Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Ігровий програмний застосунок в жанрі Shoot'em-up. Реалізація штучного інтелекту, бойової системи та системи збереження прогресу
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-7

Луценко В. В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ас. кафедри ПІ Матвєєв Д. І.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З. В. Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Луценко Віталію Володимировичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий програмний застосунок в жанрі Shoot'em-up. Реалізація штучного інтелекту, бойової системи та системи збереження прогресу.
Затверджена наказом по університету від 20.05.2024 р. № 471Ст
2. Термін подання студентом роботи до екзаменаційної комісії 10.06.2024
3. Вихідні дані до роботи Розробити ігровий програмний застосунок в жанрі Shoot'em-up, а саме систему штучного інтелекту, бойову систему та систему збереження прогресу, за допомогою ігрового рушію Unity та мови програмування C#.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	15.04.2024	*виконано*
2	Створення специфікації ПЗ	22.04.2024	*виконано*
3	Проектування ПЗ	29.04.2024	*виконано*
4	Розробка ПЗ	06.04.2024	*виконано*
5	Тестування ПЗ	13.05.2024	*виконано*
6	Оформлення пояснювальної записки	20.05.2024	*виконано*
7	Підготовка презентації та доповіді	27.05.2024	*виконано*
8	Попередній захист	06.06.2024	*виконано*
9	Нормоконтроль, рецензування	06.06.2024	*виконано*
10	Здача роботи у електронний архів	07.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	09.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка) _____ Луценко В. В.
(підпис)

Керівник роботи _____ ас. кафедри ПІ Матвєєв Д. І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Звіт з передатестаційної практики, 94 стор., 61 рис., 13 джерел. , 10 табл..

ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, СУПРОТИВНИКИ, ШІ, C#, NAVMESH, RIDER, SHMUP, SHOOT'EM UP, UNITY

Об'єкт розробки – Ігровий програмний застосунок у жанрі Shoot'em up та розробка системи штучного інтелекту для ігрового застосунку, яка відповідає потребам супротивників у іграх цього жанру.

Основною метою роботи є розробка системи ШІ, яка забезпечить інтелектуальну поведінку ворожих персонажів у грі. Система повинна бути гнучкою, щоб дозволити легке додавання нових типів противників та їхніх стратегій.

Методи розробки базуються на кодовій базі ігрового рушія Unity та мові програмування C#.

Результатом є створення демонстраційної версії ігрового застосунку у жанрі shoot'em up з естетикою sci-fi для операційної системи Windows, яка отримає назву «Dimension of Darkness». Розробка концепції абстракцій, використання, збереження та відображення поведінкових патернів. Створення прототипу системи ШІ для супротивників, реалізування тестових сценаріїв та забезпеченність можливості легкого розширення та налаштування системи.

GAME SOFTWARE APPLICATION, OPPONENTS, AI, C#, NAVMESH, RIDER, SHMUP, SHOOT'EM UP, UNITY

Object of development - A game software application in the Shoot'em up genre and the development of an artificial intelligence system for a game application that meets the needs of opponents in games of this genre.

The main goal of the work is to develop an AI system that will provide intelligent behavior of enemy characters in the game. The system should be flexible to allow easy

addition of new types of enemies and their strategies.

The development methods are based on the Unity game engine code base and the C# programming language.

The result is a demo version of a shoot'em up game application with a sci-fi aesthetic for the Windows operating system, called Dimension of Darkness. Development of the concept of abstractions, use, storage, and display of behavioral patterns. Creating a prototype of an AI system for enemies, implementing test scenarios, and ensuring the possibility of easy expansion and customization of the system.

Я, Луценко Віталій Володимирович, студент гр. ПЗПІ-20-7, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок в жанрі Shoot'em-up. Реалізація штучного інтелекту, бойової системи та системи збереження прогресу», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЕОМ – Електронно-Обчислювальна Машина

ІТ – Інформаційні технології

ПІ – Програмна Інженерія

ПЗ – Програмне Забезпечення

ХНУРЕ – Харківський Національний Університет РадіоЕлектроніки

3D – 3-Dimensional

SHMUP – Shoot’em up

NPC – Not Playable Character

ШІ – Штучний інтелект

ВСТУП

Жанр shoot 'em up – це піджанр шутерів, коріння якого можна простежити до ранніх днів відеоігор. У цьому жанрі зазвичай персонаж гравця автоматично рухається вперед і часто являє собою літаючий транспортний засіб, наприклад, літак або космічний корабель, який стріляє у ворогів, уникаючи перешкод [1]. Ворожі сили атакують з різних напрямків, стріляючи снарядами або летять в навколишньому просторі. Гравець може збирати бонуси, які підвищують потужність вогню, дають додаткові життя або інші переваги. Геймплей вимагає від гравця швидко реагувати на ворожі атаки та уникати ними.

Shoot 'em up виник у період Золотого віку відеоігор в кінці 70-х - початку 80-х років, коли аркадні ігри почали з'являтися на екранах автоматів. Початки жанру можна відслідкувати від Spacewar!, однієї з перших комп'ютерних ігор, створеної у 1961 році і випущеної для ігрових залів на початку 1970-х. Проте винахід жанру частіше приписують Томохіро Нішікадо, творцю Space Invaders. У цій грі гравець зіткнувся зі значною кількістю ворогів, які поступово збільшували швидкість руху та темп стрільби з кожним новим рівнем. Ця концепція жива і донині: у багатьох вертикальних скролл-шутерах гравець перебуває "вгорі" екрану, де постійно з'являються вороги [2].

Shoot 'em up ігри мають кілька піджанрів, які відрізняються за стилістикою, механікою гри та атмосферою. Ось деякі з найпопулярніших піджанрів: скролл-шутер, кульове пекло, біжи і стріляй, мультиспрямований шутер, тунельний шутер, рейковий шутер, фіксований шутер [1].

Ігровий застосунок, який ми розробляємо буде відноситися до піджанру мультиспрямованого шутера. Мультиспрямований шутер (англ. Multidirectional Shooter) - це піджанр shoot 'em up ігор, де гравець може керувати своїм персонажем у будь-якому напрямку, не обмежуючись лише вертикальним або горизонтальним рухом, що є однією з головних особливостей мультиспрямованих шутерів. Гравець може рухатися вгору, вниз, вліво, вправо та по діагоналі, що дозволяє йому ефективно уникати ворожих атак та стріляти в більш широкому діапазоні та стріляти по ворогам, які атакують з різних сторін. Це робить геймплей більш

динамічним та інтенсивним, вимагаючи від гравця високої швидкості реакції та керування.

Механіки бою в такому піджанрі може бути одночасно простою і досить різноманітною. Гравець може пересуватися в будь-який бік екрана, що дасть йому змогу ухилятися від безперервних ударів ворогів. Також він має різну зброю, за допомогою якої потрібно вбивати натовпи ворогів, що наступають із різних боків. Вороги теж мають різні види атаки, деякі з яких діють за шаблоном, що дає можливість гравцеві запам'ятовувати та бути готовим до кожного ворога. У багатьох іграх цього жанру зустрічаються так звані боси. Зазвичай вони більші та сильніші, ніж звичайні вороги, а також мають безліч різних атак. Щоб не програти, гравець має стежити за своїм здоров'ям, яке зменшуватиметься, коли гравець потрапить під ворожу атаку.

Цей проект спрямований на створення динамічної системи бою, де гравець має швидко реагувати і вдосконалювати свої навички керування та стрільби. Гравець матиме очки здоров'я, які не відновлюватимуться самі, і якщо гравець втратить усі очки, то він програє. Щоб збалансувати гру, у гравця буде присутній енергетичний щит, який з часом регенерується, якщо гравець не потрапляє під ворожу атаку. Це дасть гравцеві можливість довше протриматися, але для цього йому потрібно ухилятися від ворогів, які будуть швидко переміщатися і постійно атакувати ворога різними способами.

Система штучного інтелекту та супротивників в ігрових програмах є одним з ключових аспектів, який визначає рівень складності, інтелектуальності та цікавості гри. У кожного супротивника повинні бути присутніми особливі характеристики [3]. Наприклад, один швидко пересувається, маленького розміру, але має мало здоров'я і мало шкоди. Інший же навпаки, повільний, великий, має багато здоров'я, але теж має мало шкоди. А є ще один вид, який має середню кількість здоров'я, середню швидкість, але при цьому величезну шкоду. Такі характеристики супротивників змушуватимуть гравця складати власну тактику, щоб перемогти.

Зіткнувшись з усіма цими елементами бою одночасно, гравець отримає неймовірні емоції та захоплення від процесу гри.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Жанр shoot'em up є одним з найвідоміших та давно існуючих жанрів в історії відеоігор. Цей жанр виник у період розвитку аркадних ігор і з тих пір став неодмінною частиною ігрової індустрії. Shoot'em up, часто відомий як shmup, відрізняється швидким темпом гри, великою кількістю ворогів та стрілянини, а також іноді елементами босів, що робить гру динамічною та захоплюючою.

Цей жанр має велику популярність серед геймерів, особливо серед тих, хто цінує аркадну складність та адреналін від постійного стрімкого руху. Шмапи зазвичай викликають ностальгію у гравців, оскільки багато класичних шмапів стали культовими та визнаними майстерністю дизайну рівнів та геймплею.

Ринок ігор у жанрі shoot'em up є різноманітним, але менш конкурентним порівняно з іншими жанрами, такими як шутери від першої особи або рольові ігри. Однак, незважаючи на це, існує велика кількість якісних шмапів, що постійно залучають увагу фанатів жанру. Проаналізувавши деякі з таких ігор, ми зможемо зрозуміти, які аспекти потрібно взяти до уваги, щоб гра здобула хорошу репутацію, сподобалася гравцям і мала можливість розвиватися далі.

Assault Android Cactus - це ізометричний аркадний шутер з видом зверху, де гравець керує андроїдами, що борються з ордами ворожих роботів на космічному кораблі (див. рис. 1.1). Гра славиться своїм швидким темпом, динамічним геймплеєм та величезною кількістю різних андроїдів, кожен з яких має унікальні здібності. Від інших шмапів Assault Android Cactus відрізняється великою кількістю героїв з різними стилями гри, що робить кожне проходження унікальним. Гра також вражає чудовою системою супротивників, що додає динаміки, складності та інтересу до гри. Різноманітність супротивників з унікальними характеристиками і здібностями ускладнює битви: деякі противники б'ються в ближньому бою, інші тримають дистанцію, а деякі використовують статусні ефекти на гравця або інших супротивників. Гра розділена на 5 зон, кожна з яких завершується боєм з потужним босом, який володіє унікальними атаками і здібностями. Також варто відзначити прогресивне ускладнення гри: з кожним

новим рівнем противники стають складнішими та агресивнішими, не дозволяючи гравцю розслабитися. У персонажа відсутнє стандартне здоров'я, натомість він має енергію, яка витрачається щосекунди, завдяки чому гравцеві доводиться діяти швидше й агресивніше.

Рисунок 1.1 – Shoot’em up гра Assault Android Cactus (за даними [4])

Sky Force Reloaded - аркадний вертикальний шутер, де гравець управляє літаючим кораблем, суперечить великій кількості ворожих сил. Графіка Sky Force вражає своєю якістю, навіть для 2D ігри. Цей проєкт нагадує класичні shoot'em up ігри, де герой автоматично рухається вперед, часто на літаючому транспортному засобі, і відбиває атаки ворогів, уникати перешкод. Гра пропонує спільну гру для двох гравців, дозволяючи друзям або співробітникам працювати разом для досягнення спільної мети - перемоги над ворогами (див. рис. 1.2).

Система супротивників у Sky Force доволі різноманітна: у грі представлено різні типи супротивників, включно з повітряними кораблями, гелікоптерами, бомбардувальниками, наземними танками та оборонними установками. Як і в попередньому прикладі, у Sky Force є боси наприкінці рівня. Вони зазвичай більші та мають складніші атаки, ніж звичайні супротивники. Перемога над босами

вимагає хорошої стратегії та вміння ухилятися від їхніх атак. Іноді противники атакують хвилею, створюючи додаткову напругу і вимагаючи від гравця швидкої реакції та вміння стратегічно вибирати цілі для знищення. Гравець може покращувати свій літак і спорядження, що допомагає боротися з сильнішими супротивниками. Наприклад, можна покращувати зброю, швидкість і броню літака.

Рисунок 1.2 – Кооператив гри Sky Force Reloaded (за даними [5])

Серія ігор Contra - це культове творіння у жанрі run and gun, розроблене компанією Konami. Як і багато ігор цього жанру, Contra вперше з'явилася на аркадних автоматах у 1987 році [6]. Основна ідея сюжету полягає у захисті людства від інопланетних загарбників, кібернетичних створінь та інших небезпек. Гри Contra відзначалися вражаючою графікою для свого часу та можливістю грати в кооперативному режимі, що додавало іграм соціальну динаміку та змагальність. Contra суттєво вплинула на жанр "біжи і стріляй", який поєднує в собі платформер і шутер. Геймплей у грі насичений - численні вороги, що нападають з різних напрямків, стаціонарні гармати, турелі, танки, а також гранати, що вибухають. Спрайти героїв і ворогів відрізняються великим розміром і добре проробленою анімацією. Кожен рівень завершується боєвим зіткненням з унікальним "босом",

який часто має кілька фаз битви. Також у Contra присутні елементи платформера, де гравець повинен стрибати й ухилятися від перешкод та атак ворогів на різних рівнях.

1.2 Жанрові характеристики

Цей жанр охоплює кілька піджанрів, кожен з яких має свої унікальні характеристики й особливості геймплея. Ось декілька з основних піджанрів та їхні характеристики:

Vertical Shoot'em Up (вертикальні шутери):

- Вид камери. У таких іграх дії відбуваються на вертикально орієнтованому екрані, де гравець керує своїм персонажем, що рухається вгору.
- Керування. Як правило, у гравця немає вільного переміщення по екрану, він має можливість рухатися вгору, вниз, вліво і вправо.
- Вороги. Усі противники з'являються зверху екрана й атакують гравця. Гравець зі свого боку має ухилятися від їхніх пострілів і знищувати їх, стріляючи у відповідь.

Horizontal Shoot'em Up (горизонтальні шутери):

- Вид камери. Події у такому піджанрі трапляються на горизонтально орієнтованому екрані, де гравцеві доводиться контролювати власного персонажа, котрий рухається зліва направо або навпаки.
- Керування. Подібно до вертикальних шутерів, але з обмеженням на горизонтальний рух.
- Вороги. Всі супротивники з'являються з кожного боку екрана і намагаються наблизитися до персонажа, його задача залишається тією самою, як і у вертикальних шутері.

Bullet Hell (пекло куль):

- Висока складність. Такий піджанр славиться своєю дуже високою складністю, оскільки на екрані купа різних пострілів, що вимагають від гравця максимальної зосередженості та реакції.
- Унікальні патерни. У ворогів зазвичай є свої унікальні патерни, які

гравець повинен постаратися вивчити, щоб протриматися якомога довше.

– Технічні навички. Від гравця потрібні високий рівень навичок. Він має встигати ухилятися від атак ворогів, вміти вирішувати, яких ворогів вбивати перших. Усе це відбувається під постійною напругою з боку гри.

Run'n Gun (біжи і стріляй):

– Керування. У гравця є змога рухатися в будь-який бік екрана і відстрілюватися від ворогів одночасно.
– Екшн і платформинг. Цей піджанр часто комбінує між собою елементи екшену і платформінгу, що змушує гравця не тільки ухилятися і стріляти по ворогах, а й встигати долати перешкоди.

Fixed Shooter (фіксовані шутери):

– Фіксоване положення гравця. У гравця є можливість лише повертатися або пересуватися по горизонталі чи по вертикалі.
– Вороги. Атаки ворогів відбуваються з усіх напрямків, тому гравцеві потрібно швидко реагувати і знищувати супротивників.

1.3 Аналіз механік

Основні механіки, що визначають shoot'em up, включають:

– Механіка стрільби. Стрільба є основною механікою цього жанру. Вона також відображена в його назві. Гравцю потрібно активно стріляти по наближаються ворогах.
– Механіка ухилення. Ще одна з основних механік гравця є ухилення. Для того щоб протриматися в грі довше, гравець повинен рухатися в різні боки, тим самим ухиляючись від снарядів.
– Механіка босів. Зазвичай, наприкінці кожного рівня гравця чекає бос. Це противник, який більший і сильніший за інших.
– Механіка хвильових атак. Щоб гра була складнішою, існують серії ворожих атак, які зазвичай відправляються на гравця у вигляді хвиль. Це означає, що гравець буде підданий безперервному потоку ворогів протягом деякого часу.

- Механіка підвищення і апгрейди: Під час ігрової сесії, гравець може знайти бонуси або апгрейди, які збільшуватимуть його шкоду, здоров'я або швидкість. Ці бонуси можуть як просто з'являтися на карті, так і випадати з убитих супротивників.
- Рівні з різною складністю. Під час ігрової сесії, гравець може знайти бонуси або апгрейди, які збільшуватимуть його шкоду, здоров'я або швидкість. Ці бонуси можуть як просто з'являтися на карті, так і випадати з убитих супротивників.
- Система життя. Гравцеві надається обмежена кількість життів або шансів на помилку, що відновлюються певним чином (наприклад, після досягнення певного кількості балів).
- Стратегія і пам'ять. Гравцям потрібно розвивати стратегію для успішного проходження рівнів і пам'ятати шаблони атак ворогів для ефективного ухилення та контратак.

1.4 Вплив і перспективи жанру

Жанр shoot'em up сильно вплинув на інші ігри та культуру загалом. Багато ігор позичають елементи геймплею з шутерів, таких як стрільба, ухиляння від пострілів, боси і хвильові атаки. Звуковий і візуальний стиль надихнули розробників інших ігор на те, щоб повторити успіх ігор shoot'em up. Вони відомі своїми яскравими та динамічними візуальними ефектами, а музичний супровід і звукові ефекти відіграють важливу роль у створенні атмосфери та підтримці напруги. Також, слід зазначити ігрові інновації, такі як система апгрейдів або збір бонусів, які тепер можна знайти в багатьох інших жанрах.

Щодо майбутнього цього жанру, він тісно пов'язаний з розвитком технологій і смаків гравців. Сьогодні стрімко розвивається віртуальна і доповнена реальність, можливості яких можуть принести новий погляд на жанр shoot'em up. Крім того, використання штучного інтелекту і процедурної генерації рівнів може зробити геймплей різноманітнішим і захопливим, створюючи нові виклики та ситуації для гравців. Також важливим є розвиток візуального стилю і анімації, з ростом

графічних технологій ігри можуть ставати більш барвистими і захопливими, з'являючись з вражаючими візуальними ефектами.

Shoot'em up продовжує відзначатися своєю відмінною спадщиною в ігровій історії, надихаючи нові покоління розробників і гравців. Його простота та веселість дозволяють і далі залишатися популярним, навіть у світлі появи нових жанрів і технологій.

1.5 Виявлення та вирішення проблем

Виявлення та вирішення проблем у грі жанру shoot'em up може бути складним завданням, оскільки цей жанр часто вимагає від гравця швидкі реакції та точності. Ось декілька загальних проблем, які можуть виникнути у грі такого типу, а також способи їх вирішення:

- Баланс геймплею. Гра може бути занадто складною для більшості гравців, деякі ситуації у грі можуть здатися несправедливими, наприклад, коли вороги атакують гравця без жодного шансу на ухилення. Для вирішення цієї проблеми можна розглянути можливість додавання різних рівнів складності, що дозволить гравцям з різним рівнем досвіду знайти оптимальний баланс між викликом і задоволенням від гри, а також можна внести зміни до логіки ворогів або до геймплею.
- Монотонність. Якщо гра стає монотонною через повторюваність або обмежену різноманітність, можна розглянути додавання нових елементів геймплею. Наприклад, нові типи ворогів, зміна локацій або розвиток головоломок можуть додати більше різноманіття та цікавості до гри.
- Технічні проблеми. Проблеми з візуальним відтворенням, анімацією або іншими технічними аспектами можуть вплинути на загальний досвід гравців. Ці проблеми можна вирішити шляхом оновлення або поліпшення відповідних компонентів гри, щоб забезпечити їхню коректну роботу і підвищити якість гри.
- Недоліки в керуванні. Якщо керування грою не є зручним або точним, це може стати проблемою для гравців. Важливо вивчити та оптимізувати

його так, щоб воно було максимально зручним та точним.

Уникнення цих проблем вимагає не лише ретельного аналізу та пошуку оптимальних рішень, але й постійного вдосконалення гри з урахуванням відгуків гравців та технічних можливостей. Важливо забезпечити, щоб кожен аспект гри був збалансованим, цікавим та доступним для широкого кола гравців.

1.6 Постановка задачі

Для створення ігрового програмного застосунку у жанрі shoot'em up з ефективною системою штучного інтелекту та супротивників, який надасть гравцям захоплюючий і викликовий геймплей, поставлені наступні завдання:

Дослідити сучасні підходи до реалізації систем штучного інтелекту та супротивників у відеоіграх жанру shoot'em up;

Визначити основні характеристики геймплею та механіки, що характеризують жанр shoot'em up, для визначення вимог до системи штучного інтелекту;

Розробити архітектуру системи штучного інтелекту, яка б враховувала специфіку геймплею shoot'em up та забезпечувала адекватні реакції супротивників на дії гравця;

Реалізувати та інтегрувати систему штучного інтелекту та супротивників у ігровий програмний застосунок на платформі Unity;

Провести тестування системи на предмет ефективності та адаптивності до різних стилів гри гравців, забезпечивши відповідний рівень складності та викликів у геймплеї;

Результатом проекту має бути гра у жанрі shoot'em up з високоякісною системою штучного інтелекту, яка надасть гравцям захоплюючий та насичений ігровий досвід, а також дозволить дотримуватися стандартів цього жанру та конкурувати на ринку ігор.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка мети

Головною метою цієї роботи є розробка системи штучного інтелекту та супротивників для ігрового програмного застосунку у жанрі shoot'em up на платформі Unity.

У грі будуть реалізовані такі основні елементи, як система керування супротивниками, система штучного інтелекту, інтерфейс користувача з інформаційним екраном про стан гравця та супротивників, а також про поточний етап гри, стандартні механіки ворогів у стилі цього жанру.

Система штучного інтелекту та супротивників повинна бути реалізована з урахуванням взаємодії з іншими системами гри (гравець, навколишнє середовище, інші супротивники). Вона повинна забезпечувати відчуття живої та реалістичної реакції супротивників на дії гравця, тим самим створюючи динамічний геймплей. Супротивники повинні демонструвати різноманітні стратегії поведінки, включаючи атаку, ухил, розташування та використання об'єктів у середовищі. Вони також повинні мати візуальну та звукову складові, що допомагають створити атмосферу бойових дій.

2.2 Загальний опис

Система штучного інтелекту для супротивників у ігровому програмному застосунку у жанрі shoot'em up представляє собою набір алгоритмів та правил, що визначають поведінку ворожих сил у грі. Вона включає в себе алгоритми навігації, атаки та захисту супротивників, а також механізми для підтримки імітації реальної стратегічної поведінки.

Крім того, вороги повинні мати анімації, які відображатимуть їхні дії. Супротивники можуть мати різну швидкість ходіння або бігу. Деякі можуть пересуватися стрибками. Окрім того, мають бути різні анімації атак, як для ближнього бою так і для дальнього. При отриманні шкоди або смерті, ворог має бути анімований відповідно.

Ця система спроектована таким чином, щоб забезпечити ворогам відповідну

реакцію на дії гравця та створити відчуття інтелектуальної гри. Алгоритми навігації дозволяють супротивникам ефективно рухатися по ігровому полю, уникати перешкод та знаходити оптимальний шлях до гравця. Алгоритми атаки та захисту враховують різноманітні стратегії бойових дій, включаючи вогневу потужність, тактику приховування та спроби уникнути вогню гравця.

Крім того, система штучного інтелекту передбачає підтримку динамічної зміни стратегій супротивників у залежності від обстановки на полі бою. Що довше гравцеві вдається протриматися живим у грі, то сильнішими ставатимуть противники. Така система забезпечить інтерес гравця до гри, оскільки він прагнутиме отримати найкращий результат. Це дасть змогу грі розвиватися й отримувати нову аудиторію.

2.3 Загальні обмеження

Як і ігровий застосунок – система штучного інтелекту та супротивників має бути реалізований на ігровому рушії Unity3D версії 2022.3.8f1 з використанням мов програмування C#. Система штучного інтелекту повинна бути реалізована за допомогою технології навігації NavMeshAgent, а також створених вручну методів для атаки.

Мінімальні системні вимоги ігрового застосунку:

- Процесор: Intel Core i7-10700K / AMD Ryzen 7 5800X.
- ОЗП: 16 GB RAM.
- Відеокарта: AMD Radeon RX 5600 XT / NVIDIA GeForce GTX 1660 Super.
- Місце на диску: 60 GB на SSD

Система має працювати на операційній системі Windows 10 або 11 та версії ОС x64.

2.4 Припущення та залежності

У проекті розглядаються наступні припущення та залежності:

- ігровий програмний застосунок може бути запущений на платформі Windows 10 або 11. Дане обмеження обумовлено використанням

функціоналу та можливостей, що доступні тільки у вищих версіях операційної системи Windows, а також для забезпечення сумісності з іншими компонентами гри;

- ігровий програмний застосунок може бути запущений на комп'ютері, що не відповідає мінімальним системним вимогам. Однак, для оптимальної роботи гри рекомендується використання комп'ютера, який відповідає хоча б мінімальним системним вимогам;
- для того, щоб система штучного інтелекту супротивників працювала, потрібна робоча система керуванням гравцем, з яким буде взаємодіяти система ворогів;
- система навігації супротивників може мати проблеми з роботою на інших версіях рушія.

Система ШІ та супротивників використовує компоненти NavMesh, версія котрих може не підтримуватися на версії нижче або вище за рушій (2022.3.8f1).

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Для візуалізації опису функціональних можливостей системи супротивників з точку зору користувача, можна скористатися діаграмою прецедентів.

Для нашого ігрового застосунку ми створили діаграму прецедентів, яка відображає дії ворога у грі (див. рис. 3.1).

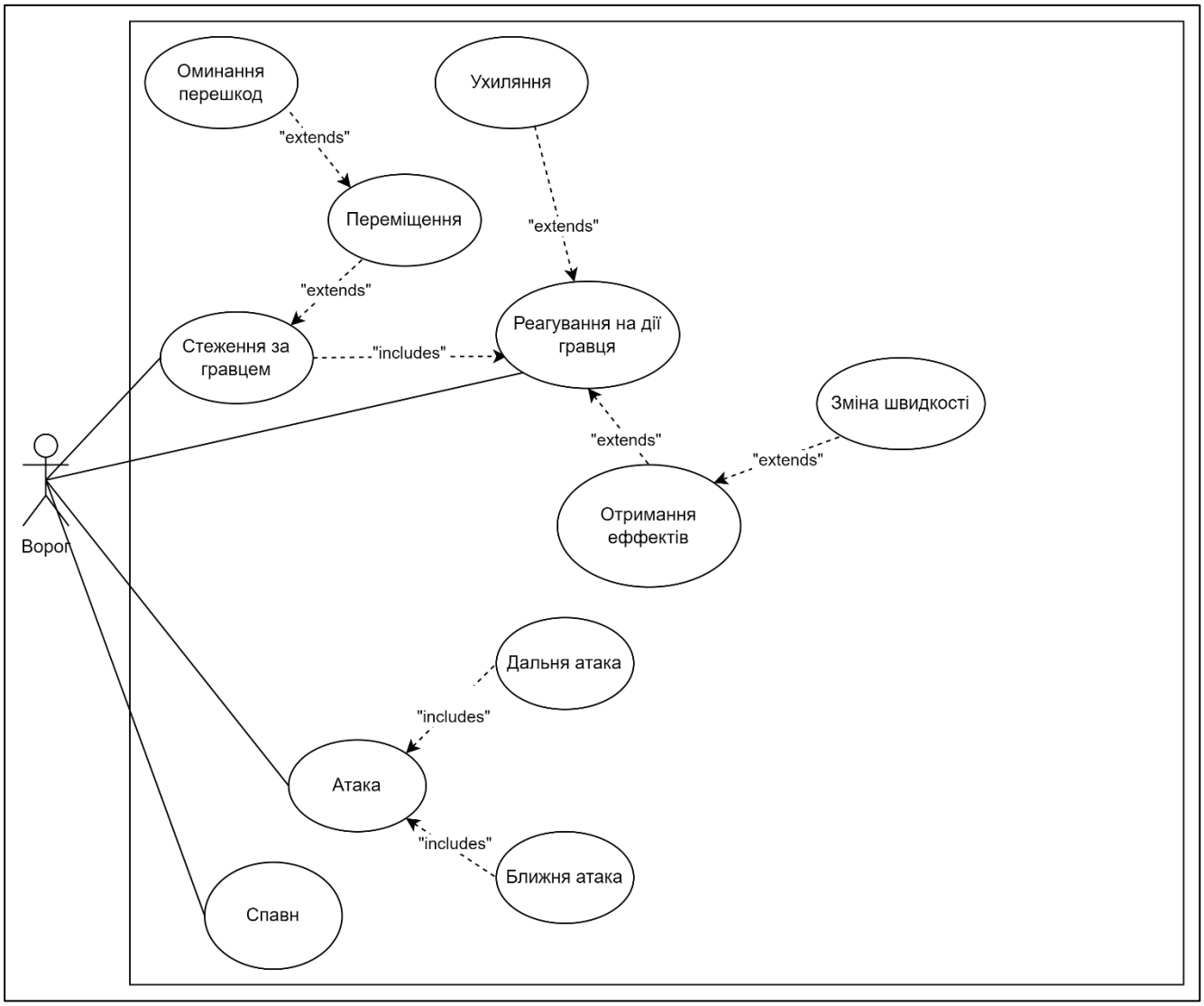


Рисунок 3.1 – Діаграма прецедентів (рисунок виконаний самостійно)

За допомогою цієї діаграми можна спостерігати за взаємодією між ігровим додатком і ворогами, де описані основні дії, які можуть бути виконані ворогами. Такий тип діаграми допомагає у розробці та проектуванні системи керування супротивниками та визначити їх основні функції у ігровому застосунку [7].

3.2 Проєктування архітектури ПЗ

Діаграма класів є важливою частиною архітектурного проекту нашого ігрового застосунку жанру shoot 'em up. Ця діаграма відображає структуру класів і їх взаємозв'язки, необхідні для реалізації системи керування противниками в грі [8]. Розробка цієї діаграми допоможе нам краще розібратися в логіці гри та забезпечити її ефективне функціонування. (див. рис. 3.2)

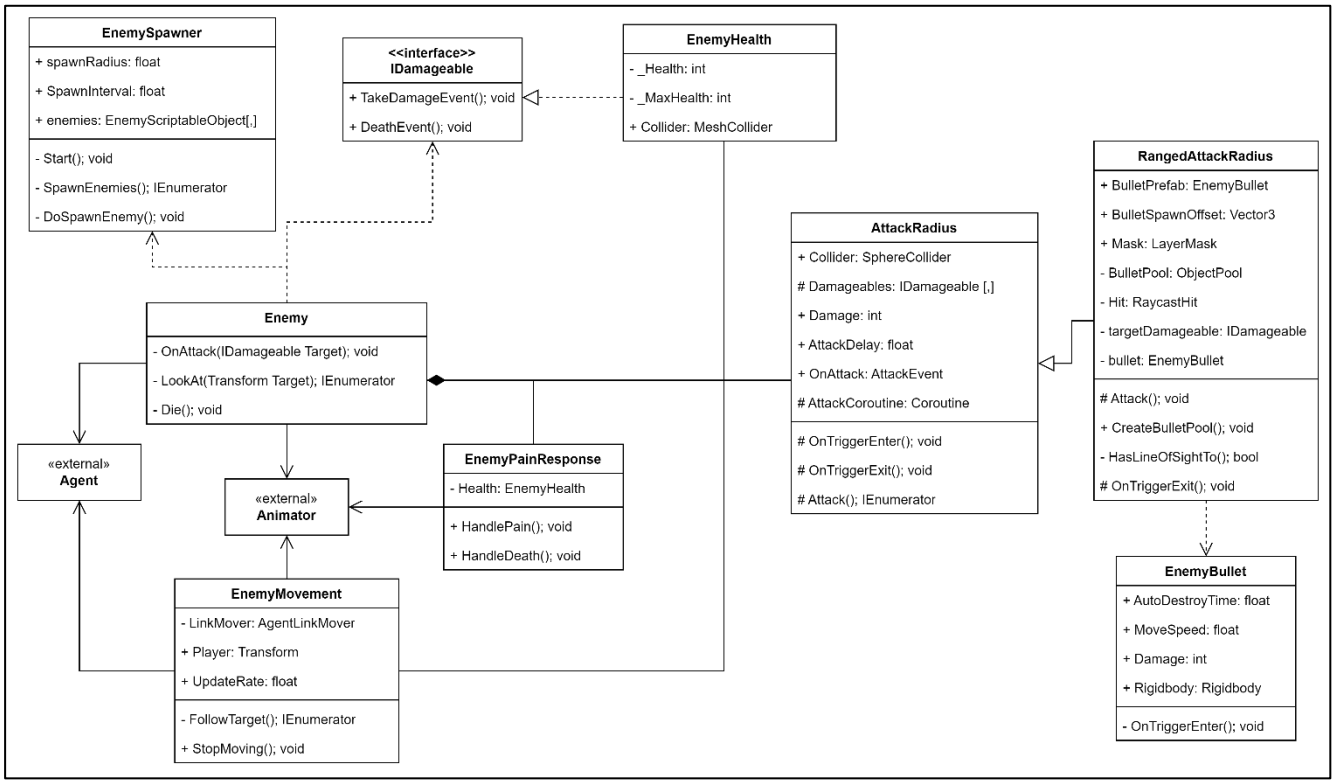


Рисунок 3.2 – Діаграма класів, пов’язаних з супротивниками(рисунок виконаний самостійно)

На діаграмі представлені основні класи, які відповідають за різні аспекти поведінки та керування противниками в грі. Кожен клас має свої властивості та методи, що визначають його функціональність. Також у діаграмі відображені зв'язки між класами, що допомагають зрозуміти логіку взаємодії між ними.

Дана діаграма є базовим каркасом для подальшої реалізації системи керування противниками в грі. Вона може бути доповнена та модифікована в процесі розробки для врахування конкретних потреб та вимог проекту.

3.3 Приклади найцікавіших алгоритмів та методів

Найцікавішими алгоритмами у грі є алгоритми навігації та алгоритм атаки супротивників.

Діаграма станів навігації NavMesh відображає процес переміщення противників у грі від пошуку шляху до досягнення цілі. Ця діаграма відображає ключові стани та переходи між ними, які відбуваються під час навігації [9]. (див. рис. 3.3).

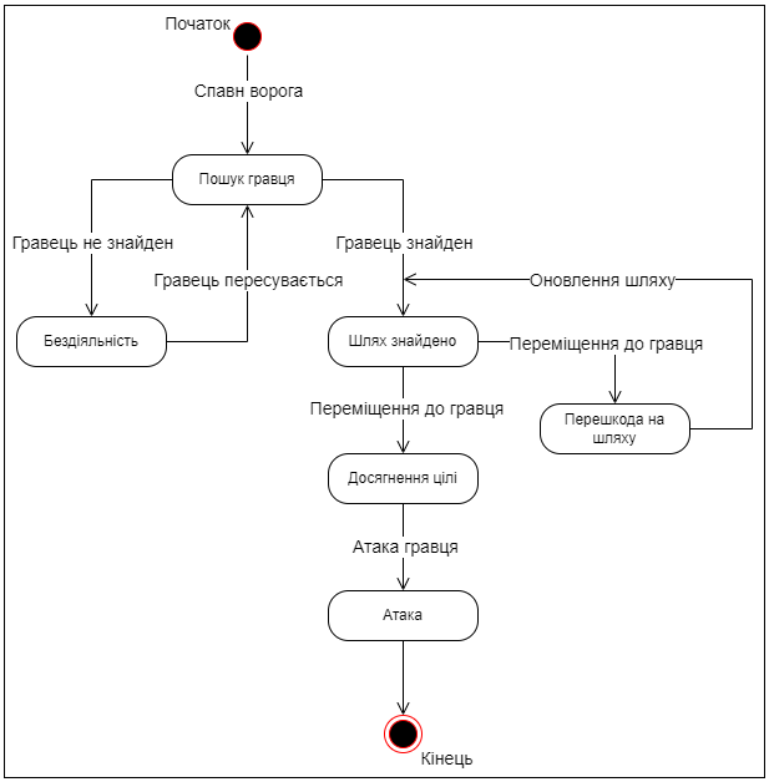


Рисунок 3.3 – Діаграма станів для навігації (рисунок виконаний самостійно)

Противники в грі повинні здати визначити оптимальний шлях до цілі, уникати перешкод та ефективно діяти під час досягнення цілі. Алгоритм пошуку шляху працює таким чином:

а) Якщо гравець у полі зору ворога, то ворог прямує до гравця.

б) Якщо гравець не у полі зору ворога, то ворог не діє.

в) Якщо на шляху ворога перешкода, то він шукає як обійти цю перешкоду.

г) Якщо ворог досяг цілі, то він атакує.

Алгоритм навігації NavMesh є важливою складовою сучасних ігрових

систем, що дозволяє створювати ефективні та реалістичні шляхи для руху об'єктів, таких як противники [10].

Наступним алгоритмом є алгоритм поведінки атаки супротивників у грі (див. рис. 3.4).

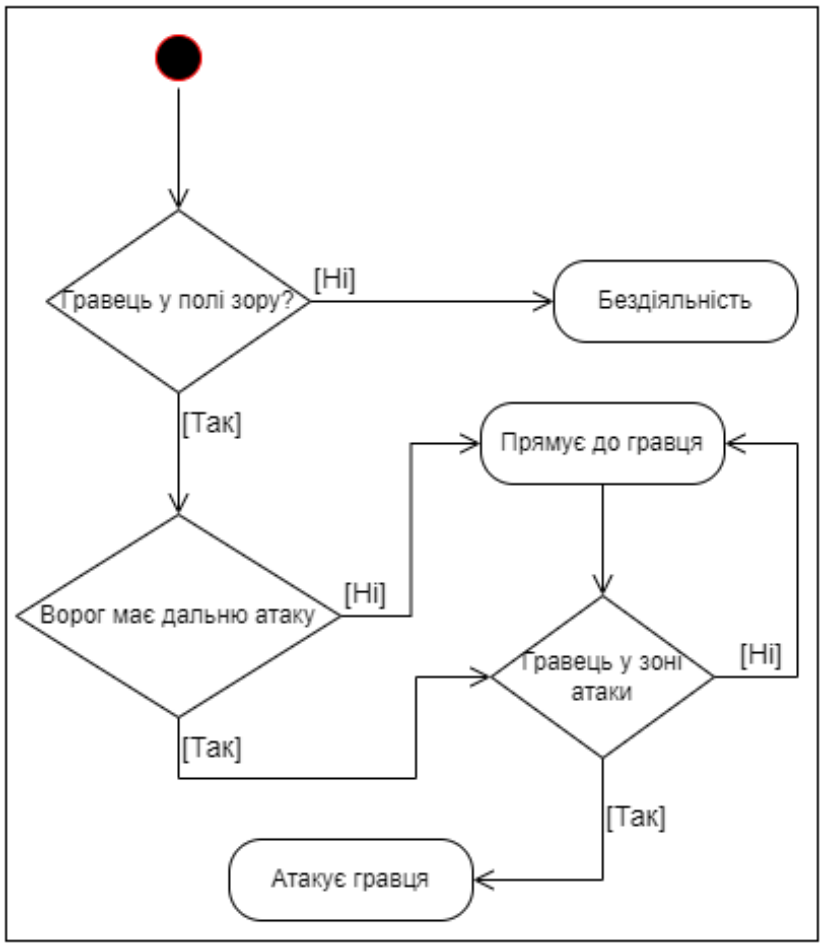


Рисунок 3.4 – Діаграма активності для поведінки атаки супротивників (рисунок виконаний самостійно)

Алгоритм атаки полягає у наступному:

а) Якщо ворог має дальню атаку, гравець у його полі зору і у зоні атаки, тоді ворог атакує;

б) Якщо ворог не має дальню атаку та гравець у його полі зору, тоді ворог прямує до гравця, і коли гравець буде у зоні атаки – ворог атакує;

в) Якщо гравець не у полі зору ворога, тоді ворог не діє;

Алгоритм атаки дуже простий, але його реалізація дозволить нам підвищити рівень виклику для гравця та забезпечити напружений та цікавий гемплей.

3.4 Створення дизайну супротивників

Дизайн супротивників відіграє ключову роль у створенні атмосфери гри та у формуванні ігрового досвіду гравця. Створення дизайну в шутемап грі вимагає ретельного планування та розробки, оскільки їхній вигляд, поведінка та інтеракція з гравцем безпосередньо впливають на загальне враження від гри.

Стиль ворогів у нашій грі має відповідати тематиці dark sci-fi. Основна ідея гри полягає в тому, що гравець опиняється на невідомій планеті, де мешкають раніше не відкриті людству форми життя. Завдання гравця - вижити серед численних небезпек, що його оточують. Тому зовнішній вигляд ворогів повинен викликати страх або тривогу, створюючи напружену атмосферу. Вороги мають виглядати загрозливо, щоб гравець відчував постійне відчуття небезпеки та викликів на кожному кроці.

Вид моделі має відповідати характеристикам ворога. Наприклад, є ворог із двома довгими лапами, за допомогою яких він переміщається й атакує, а також не має будь-якого захисту на тілі (див. рис. 3.5). За таким описом можна зрозуміти, що ворог матиме невелику кількість здоров'я і переміщатиметься з повільною швидкістю.

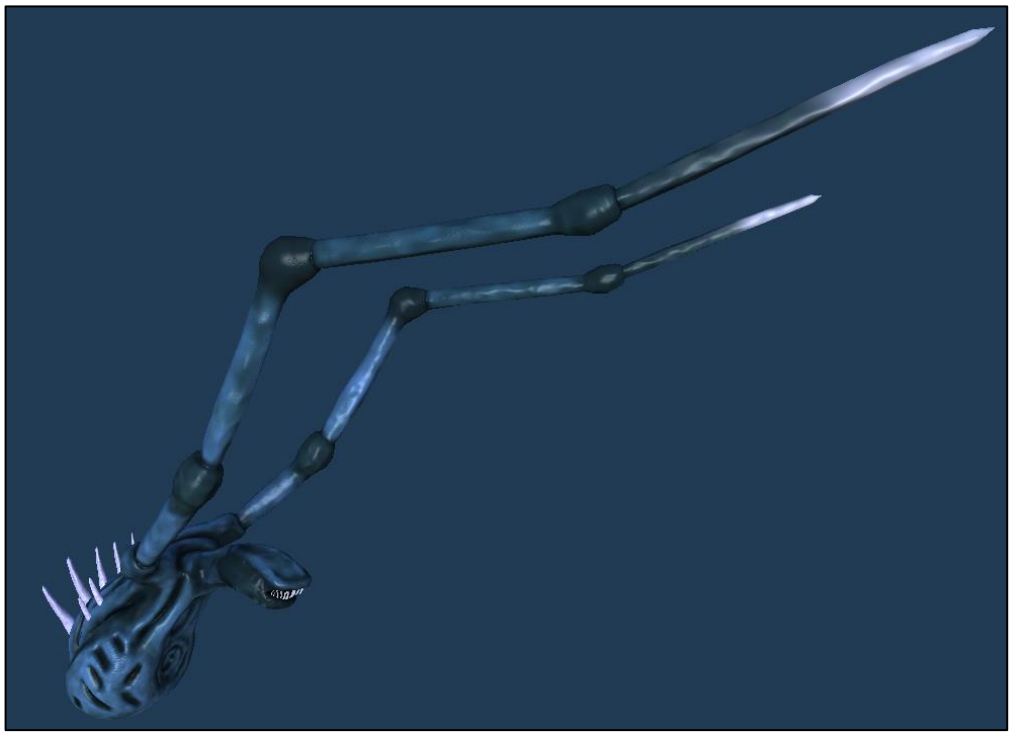


Рисунок 3.5 – Модель першого ворога (рисунок виконаний самостійно)

Наступна модель противника має панцир на тілі, 4 лапи для пересування і 2 для атаки, а отже, у ворога буде більше здоров'я і вища швидкість пересування (див. рис. 3.6).

Рисунок 3.6 – Модель другого ворога (рисунок виконаний самостійно)

Остання модель має менший розмір, ніж у попередніх. У неї 6 рук у вигляді лап і язик, яким ворог атакуватиме. Маленький розмір означає маленьку кількість здоров'я, але швидкість пересування може бути збільшена (див. рис. 3.7).

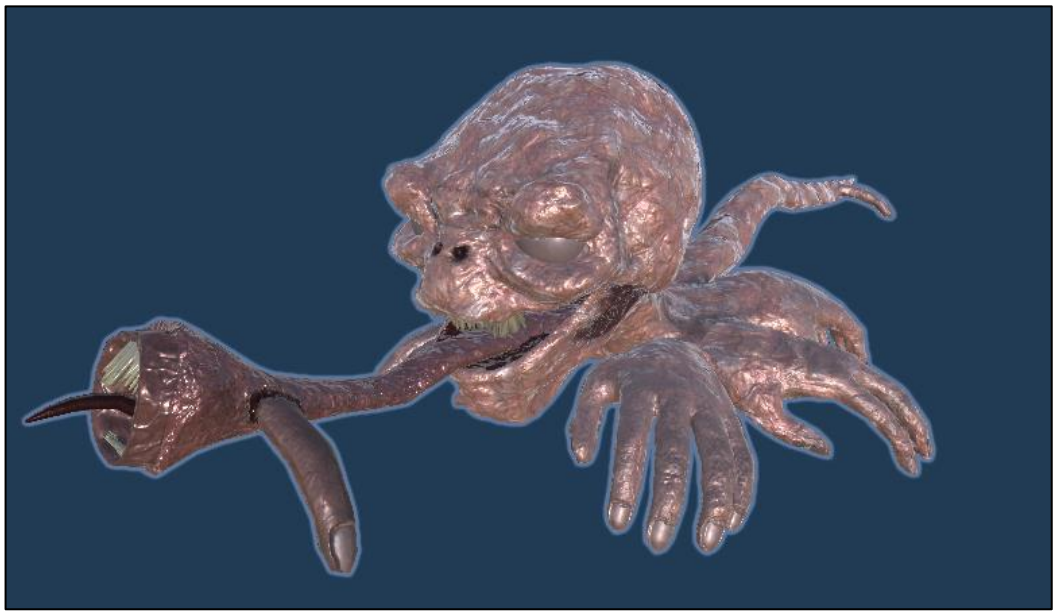


Рисунок 3.7 – Модель третього ворога (рисунок виконаний самостійно)

Також було ухвалено рішення змінювати модель противника залежно від його виду. Для прикладу було створено 3 різновиди ворогів: ближнього бою, дальнього бою і посилений вид супротивників (див. рис. 3.8). Супротивники відрізняються за кольором та розміром.

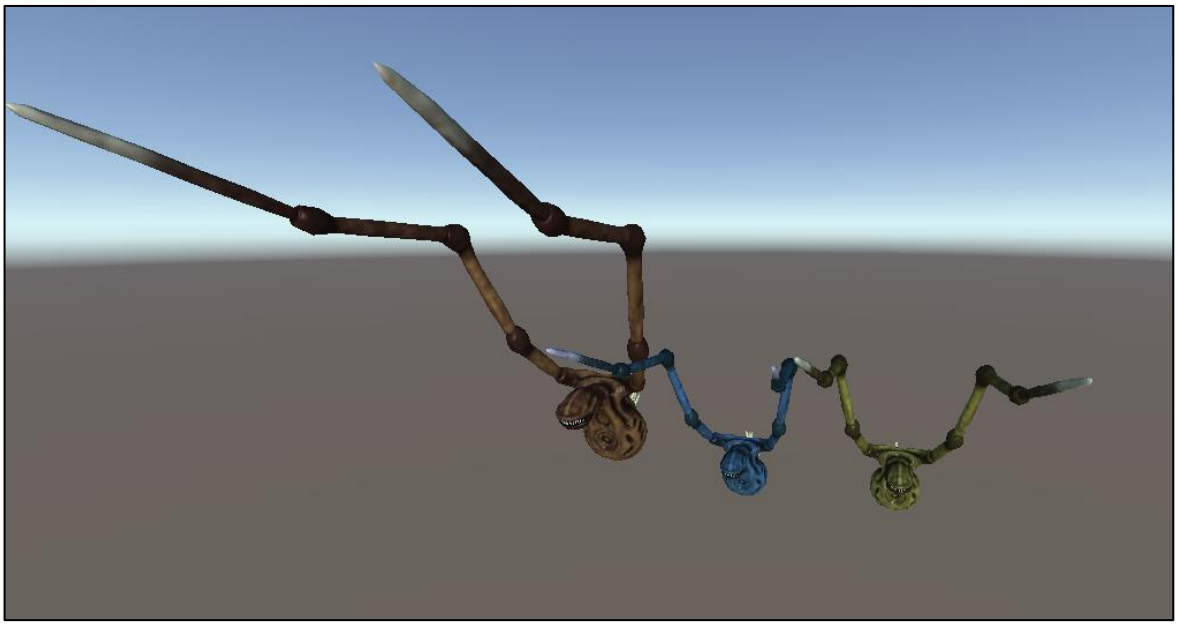


Рисунок 3.8 – Різновиди ворогів (рисунок виконаний самостійно)

Розглянемо анімації супротивників. Кожен ворог повинен мати стандартний набір анімацій: пересування, атака, отримання шкоди, смерть (див. рис. 3.9). В Unity для цього використовується компонент Animator, який дає змогу створювати переходи між станами.

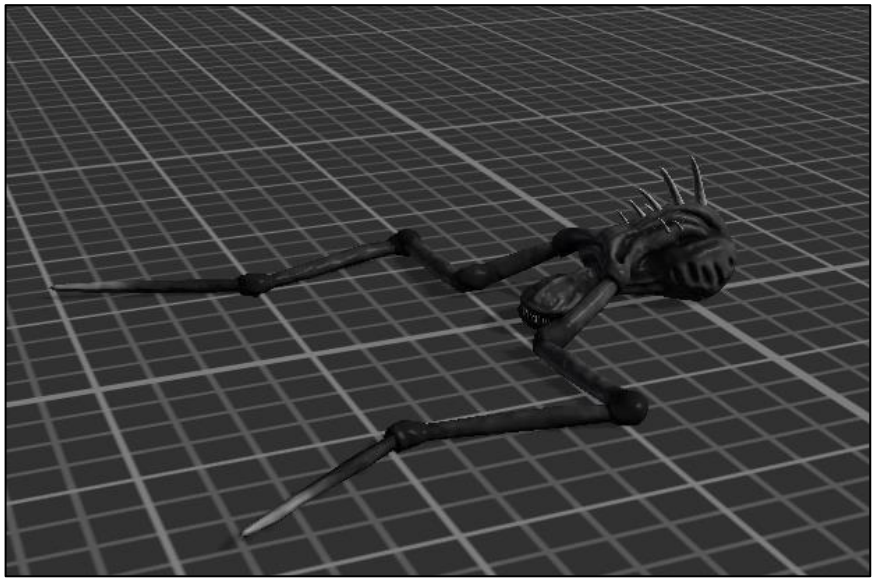


Рисунок 3.9 – Анімація смерті противника (рисунок виконаний самостійно)

Для реалізації переходу з будь-якого стану в стан атаки було використано Animation Layers [11] (див. рис. 3.10). В основному шарі створено анімації пересування, отримання шкоди і смерті. Також є анімація очікування, але поки гравцеві не видно її.

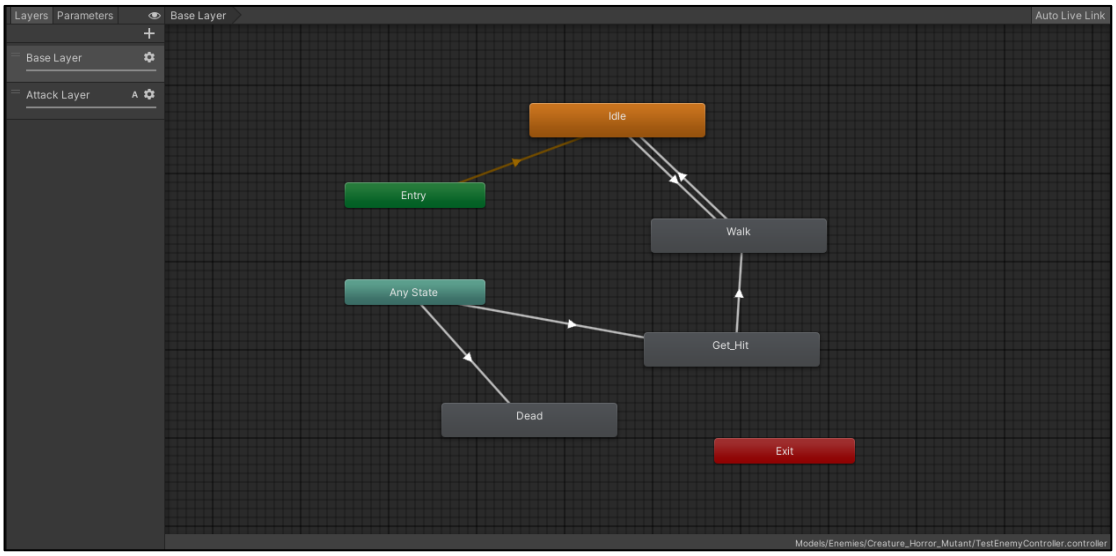


Рисунок 3.10 – Base Layer аніматора (рисунок виконаний самостійно)

Другий шар відповідає за атаку противника. Ворог може атакувати з будь-якого стану, крім смерті (див. рис. 3.11).

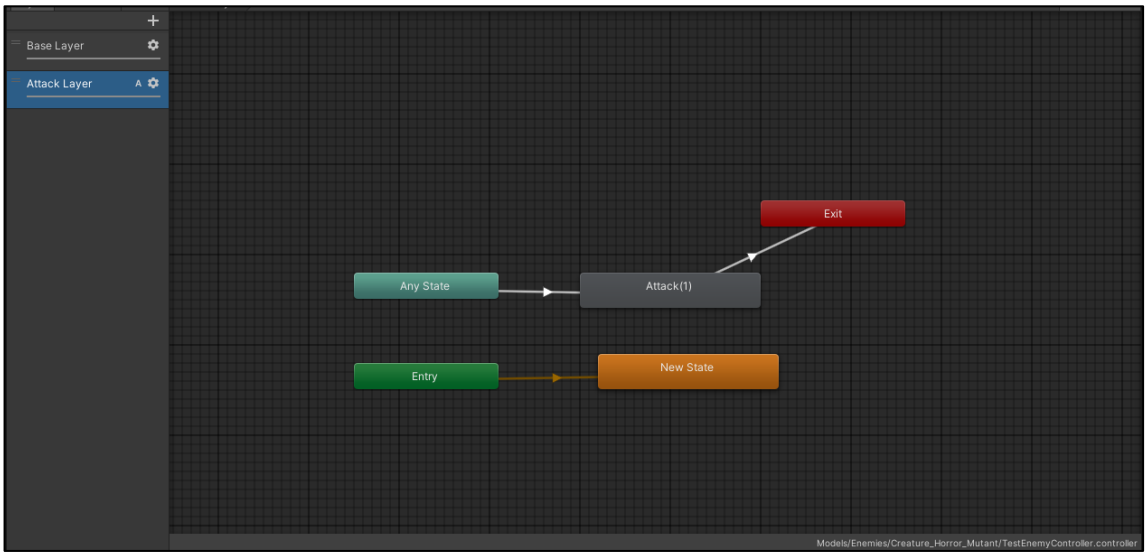


Рисунок 3.10 – Attack Layer аніматора (рисунок виконаний самостійно)

Слої анімації дають змогу створювати складніші анімаційні стани, об'єднуючи різні частини тіла в одному контролері анімації.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Налаштування параметрів супротивників

Для налаштування параметрів супротивників було вирішено використовувати ScriptableObject [12]. За допомогою такого підходу процес розроблення гри стає зручнішим і гнучкішим, даючи змогу в будь-який момент змінити параметри ворога без необхідності переписувати код. Крім того, ScriptableObject дає змогу створювати конфігураційні файли, які можна редагувати прямо в інспекторі Unity, що значно спрощує налаштування і тестування гри.

В EnemyScriptableObject задаються параметри супротивника, як-от кількість здоров'я, і налаштування агента NavMesh, як-от швидкість, прискорення, швидкість повороту та інші:

```
        // Enemy Stats
        public int Health = 100;

        // NavMesh Configs
        public float AIUpdateInterval = 0.1f;
        public float Acceleration = 8;
        public float AngularSpeed = 200;
        public int AreaMask = -1;
        public int AvoidancePriority = 50;
        public float BaseOffset;
        public float Height = 2f;
        public     ObstacleAvoidanceType     ObstacleAvoidanceType     =
ObstacleAvoidanceType.LowQualityObstacleAvoidance;
        public float Radius = 0.5f;
        public float Speed = 3f;
        public float StoppingDistance = 0.5f;
```

Далі метод SetupEnemy налаштовує всі параметри для противника, а також викликає такий самий метод з AttackScriptableObject для налаштування параметрів атаки:

```
    public void SetupEnemy(Enemy enemy)
        {
            enemy.Agent.acceleration = Acceleration;
            enemy.Agent.angularSpeed = AngularSpeed;
            enemy.Agent.areaMask = AreaMask;
            enemy.Agent.avoidancePriority = AvoidancePriority;
            enemy.Agent.baseOffset = BaseOffset;
            enemy.Agent.height = Height;
```

```
            enemy.Agent.obstacleAvoidanceType = ObstacleAvoidanceType;
            enemy.Agent.radius = Radius;
            enemy.Agent.speed = Speed;
            enemy.Agent.stoppingDistance = StoppingDistance;

            enemy.Movement.UpdateRate = AIUpdateInterval;

            enemy.Health._MaxHealth = Health;

            AttackConfiguration.SetupEnemy(enemy);
        }
```

У конфігураційному файлі AttackScriptableObject так само, як і в попередньому, спочатку задаються параметри для атаки, а після викликається метод SetupEnemy, який налаштовує параметри атаки, і якщо атака дистанційна – у налаштування входять додаткові параметри:

```
    public bool isRanged = false;
            public int Damage = 5;
            public float AttackRadius = 1.5f;
            public float AttackDelay = 1.5f;

            // Ranged Configs
            public EnemyBullet BulletPrefab;
            public Vector3 BulletSpawnOffset = new Vector3(0, 1, 0);
            public LayerMask LineOfSightLayers;

            public void SetupEnemy(Enemy enemy)
            {
                enemy.AttackRadius.Collider.radius = AttackRadius;
                enemy.AttackRadius.AttackDelay = AttackDelay;
                enemy.AttackRadius.Damage = Damage;

                if (isRanged)
                {
                    RangedAttackRadius        rangedAttackRadius        =
enemy.AttackRadius.GetComponent<RangedAttackRadius>();

                    rangedAttackRadius.CreateBulletPool();
                    rangedAttackRadius.BulletPrefab = BulletPrefab;
                    rangedAttackRadius.BulletSpawnOffset                =
BulletSpawnOffset;
                    rangedAttackRadius.Mask = LineOfSightLayers;


                }
            }
```

Одну й ту саму конфігурацію можна використовувати для кількох супротивників, і в майбутньому можна легко додавати нові типи атак і

супротивників, наприклад супротивник з атакою ближнього бою (див. рис. 4.1) та с дальньою атакою (див. рис. 4.2).

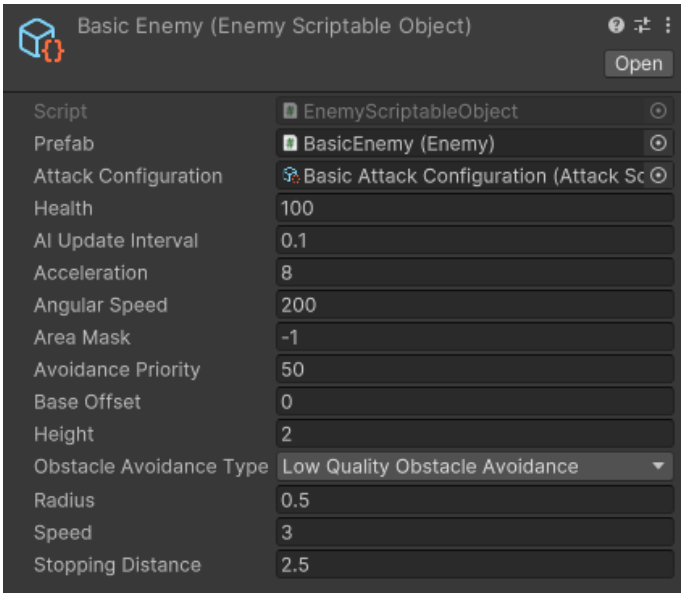


Рисунок 4.1 – Конфігурація ближнього бою (рисунок виконаний самостійно)

Тут можна помітити, що структура двох супротивників однакова, але їхні характеристики відрізняються.

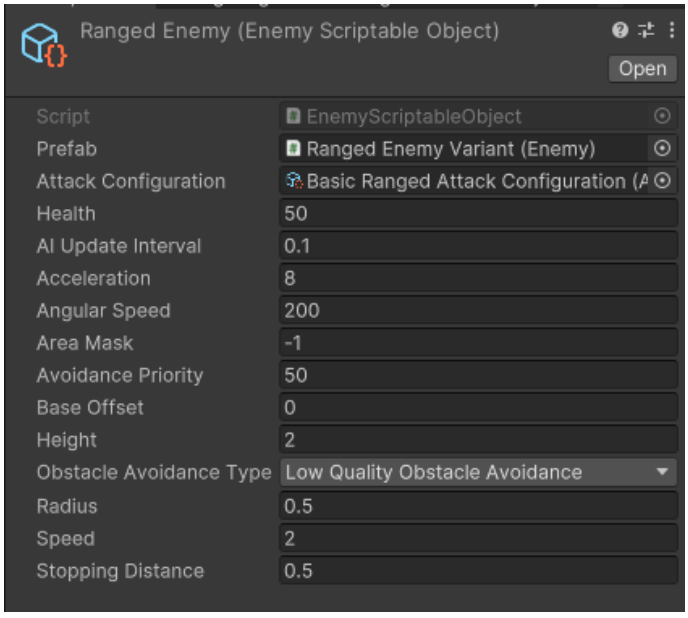


Рисунок 4.2 – Конфігурація дальнього бою (рисунок виконаний самостійно)

Отже, така система дає змогу економити час на тестування і робить налаштування балансу гри в рази зручнішим.

4.2 Навігація та атака супротивників

Навігація супротивників реалізована за допомогою системи NavMesh, яка дає можливості для зручного керування рухом супротивників. Завдяки системі навігації NavMesh, можна створювати маршрути для ворогів, прораховуючи всі перешкоди і складності рельєфу. Для цього, у кожного ворога потрібен бути компонент Nav Mesh Agent (див. рис. 4.3).

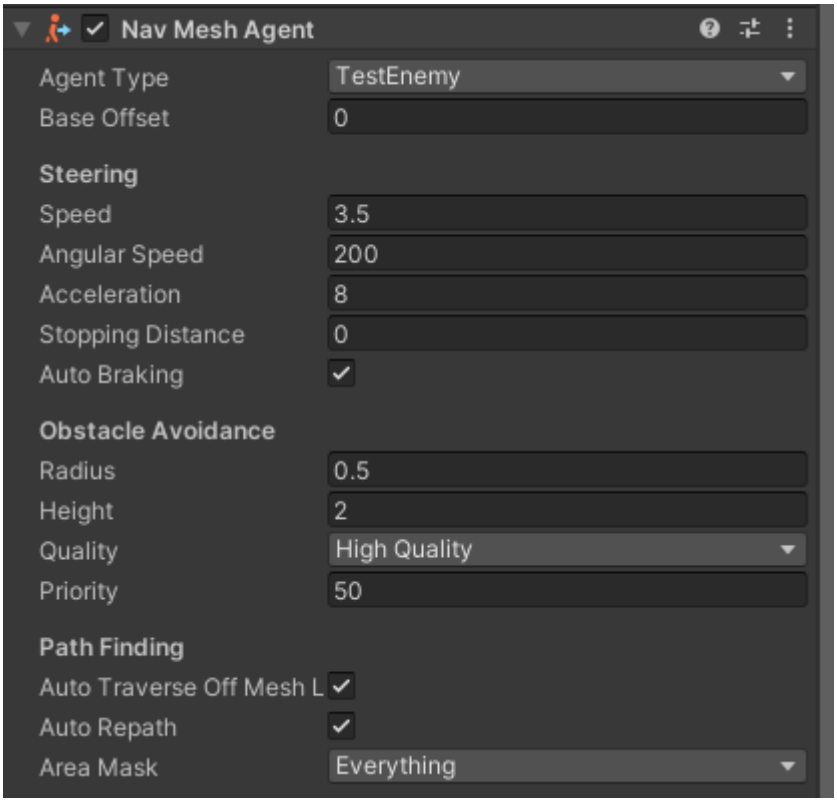


Рисунок 4.3 – Компонент Nav Mesh Agent (рисунок виконаний самостійно)

Якщо гравець живий і агент NavMesh увімкнений, тоді запускається корутина відстеження гравця:

```
        private void Start()
        {
            if (Player != null && Agent.enabled)
            {
                StartCoroutine(FollowTarget());
            }
            else
            {
                Debug.LogError("Player Transform is not assigned or
NavMeshAgent is not enabled in EnemyMovement.");
            }
```

```
    }
```

Код корутини:

```
private IEnumerator FollowTarget()
    {
        var Wait = new WaitForSeconds(UpdateRate);

        while (enabled)
        {
            if (Player != null && Agent.isOnNavMesh)
            {
                Agent.SetDestination(Player.position);
            }

            yield return Wait;
        }
    }
```

Для реалізації атаки використовуються два класи AttackRadius і RangedAttackRadius. У кожного ворога існує радіус атаки (див. рис. 4.4) , коли ціль заходить у цей радіус, то вона додається до списку:

```
        protected virtual void OnTriggerEnter(Collider other)
        {
            if (!other.TryGetComponent<IDamageable>(out var damageable))
return;
            Damageables.Add(damageable);

            StartAttack();
        }
```

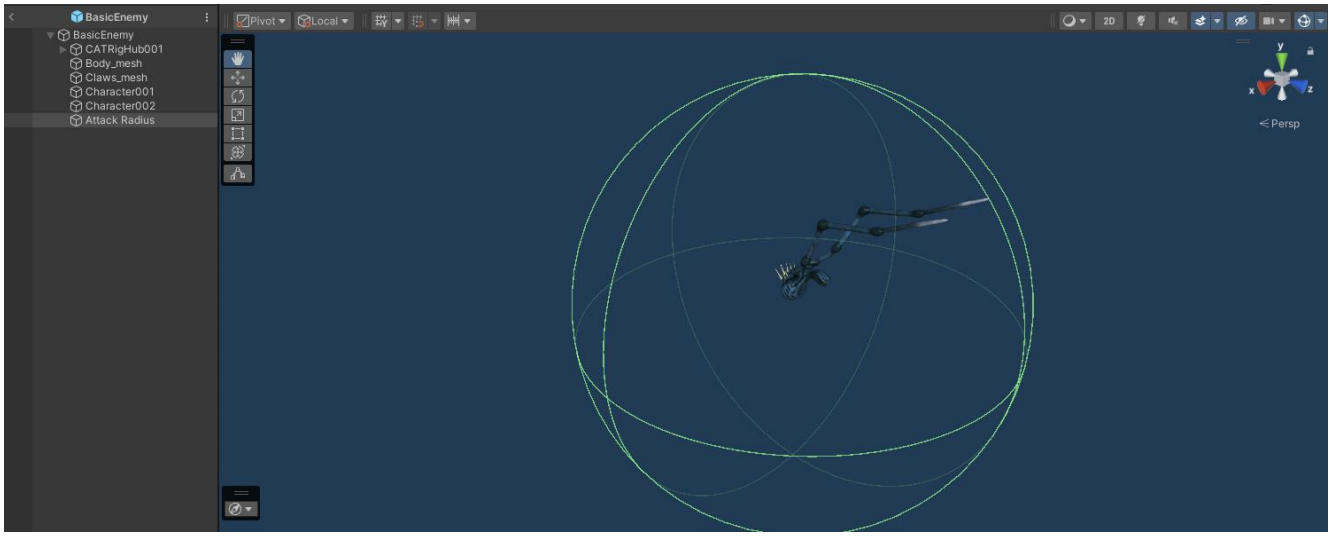


Рисунок 4.4 – Радіус атаки ворога (рисунок виконаний самостійно)

Після цього запускається атака найближчої цілі:

```
        protected virtual IEnumerator Attack()
        {
            var Wait = new WaitForSeconds(AttackDelay);
            IDamageable closestDamageable = null;
            var closestDistance = float.MaxValue;

            while (Damageables.Count > 0)
            {
                for (var i = 0; i < Damageables.Count; i++)
                {
                    var damageableTransform = Damageables[i].GetTransform();
                    var distance = Vector3.Distance(transform.position, damageableTransform.position);

                    if (distance < closestDistance)
                    {
                        closestDistance = distance;
                        closestDamageable = Damageables[i];
                    }
                }

                if (closestDamageable != null)
                {
                    OnAttack?.Invoke(closestDamageable);
                    closestDamageable.TakeDamage(Damage);
                }

                closestDamageable = null;
                closestDistance = float.MaxValue;

                yield return Wait;

                Damageables.RemoveAll(DisabledDamageables);
            }
            AttackCoroutine = null;
        }
```

Якщо ціль виходить із радіуса атаки, вона видаляється зі списку і зупиняє атаку:

```
        protected virtual void OnTriggerExit(Collider other)
        {
            if (!other.TryGetComponent<IDamageable>(out var damageable)) return;
            Damageables.Remove(damageable);
            if (Damageables.Count != 0) return;
            StopAttack();
```

```
        }

        public void StopAttack()
        {
            if (AttackCoroutine != null)
                StopCoroutine(AttackCoroutine);
            AttackCoroutine = null;
        }
```

Дальня атака працює за таким самим принципом, але перед атакою йде перевірка, чи є якісь перешкоди між ворогом і гравцем:

```
        private bool HasLineOfSightTo(Transform Target)
        {
            if (Physics.SphereCast(transform.position + BulletSpawnOffset,
SpherecastRadius,
                    (Target.position      +      BulletSpawnOffset      -
(transform.position + BulletSpawnOffset)).normalized, out Hit,
                    Collider.radius, Mask))
            {
                IDamageable damageable;
                if  (Hit.collider.TryGetComponent(out  damageable))  return
damageable.GetTransform() == Target;
            }
            return false;
        }
```

Для дальніх атак використовується снаряд, який береться з пулу куль:

```
        public void CreateBulletPool()
        {
            if (BulletPool == null)
            {
                BulletPool                                                  =
ObjectPool.ObjectPool.CreateInstance(BulletPrefab,
                    Mathf.CeilToInt(1        /        AttackDelay        *
BulletPrefab.AutoDestroyTime));
            }
        }
```

BulletPrefab: це префаб об'єкта кулі, який буде використовуватися для створення екземплярів куль у кулі. Тобто снарядом може бути будь-яка фігура або модель, а також можна зробити кілька видів снарядів.

4.3 Отримання шкоди противниками

Кожен об'єкт, який здатний отримувати шкоду, повинен мати інтерфейс IDamageable, що визначає основні методи та події:

```
public interface IDamageable
{
    public delegate void DeathEvent(Vector3 Position);
    public delegate void TakeDamageEvent(int Damage);
    public int CurrentHealth { get; }
    public int MaxHealth { get; }
    public event TakeDamageEvent OnTakeDamage;
    public event DeathEvent OnDeath;
    public void TakeDamage(int Damage);
    Transform GetTransform();
}
```

За керування здоров'ям супротивників відповідає клас EnemyHealth, а саме його метод TakeDamage:

```
public void TakeDamage(int Damage)
{
    var damageTaken = Mathf.Clamp(Damage, 0, CurrentHealth);
    CurrentHealth -= damageTaken;
    if (damageTaken != 0) OnTakeDamage?.Invoke(damageTaken);
    if (CurrentHealth <= 0 && damageTaken != 0)
    {
        OnDeath?.Invoke(transform.position);
    }
}
```

Клас EnemyPainResponse відповідає за опрацювання анімаційних реакція на отриману шкоду і смерть:

```
public void HandlePain(int Damage)
{
    if (Health.CurrentHealth > 0)
    {
        Animator.SetTrigger("Hit");
    }
}
public void HandleDeath()
{
    Animator.applyRootMotion = true;
    Animator.SetTrigger("Die");
    Animator.SetBool("isDead",true);
}
```

Така система отримання шкоди забезпечує ефективне керування процесом та дозволяє розширювати функціонал при необхідності.

4.4 Рахунок гравця та ускладнення супротивників

У цій грі рахунок гравця представлений очками темної енергії, які гравець отримує за знищення супротивників або просто кожну секунду прожиту в грі. Вони є основним показником прогресу гравця і впливають на складність гри.

Для керування очками в грі використовується клас PointsManager. Цей клас відповідає за накопичення очок та їх відображення на екрані. Код методу за допомогою якого накопичуються очки:

```
        public void RpcAddDarkEnergyPoints(int points)
        {
            darkEnergyPoints += points;
            if (darkEnergyPoints >= darkEnergyPointsToAccess)
            {
                darkEnergyPointsToAccess = firstFibonacci + 
secondFibonacci;
                secondFibonacci = firstFibonacci;
                firstFibonacci = darkEnergyPointsToAccess;
                counterICanEnter += 1;
            }
        }
```

Якщо кількість очок темної енергії досягає або перевищує значення, відбувається оновлення значення на наступне число в послідовності Фібоначчі за формулою 4.1:

$$D(n+1) = F(n) + F(n-1) = F(n+1) \quad (4.1)$$

де D – darkEnergyPointsToAccess;

n – номер кроку;

$F(n)$ – firstFibonacci;

$F(n-1)$ – secondFibonacci.

Складність ворогів безпосередньо залежить від кількості накопичених гравцем очок темної енергії. Для цього використовується спеціальний скриптовий об’єкт ScalingScriptableObject:

```
public class ScalingScriptableObject : ScriptableObject
{
    public AnimationCurve HealthCurve;
    public AnimationCurve DamageCurve;
    public AnimationCurve SpeedCurve;
    public AnimationCurve SpawnRateCurve;
    public AnimationCurve SpawnCountCurve;
}
```

Цей об’єкт дозволяє зробити криву анімації, за якою буде збільшуватися характеристика противника (див. рис. 4.5).

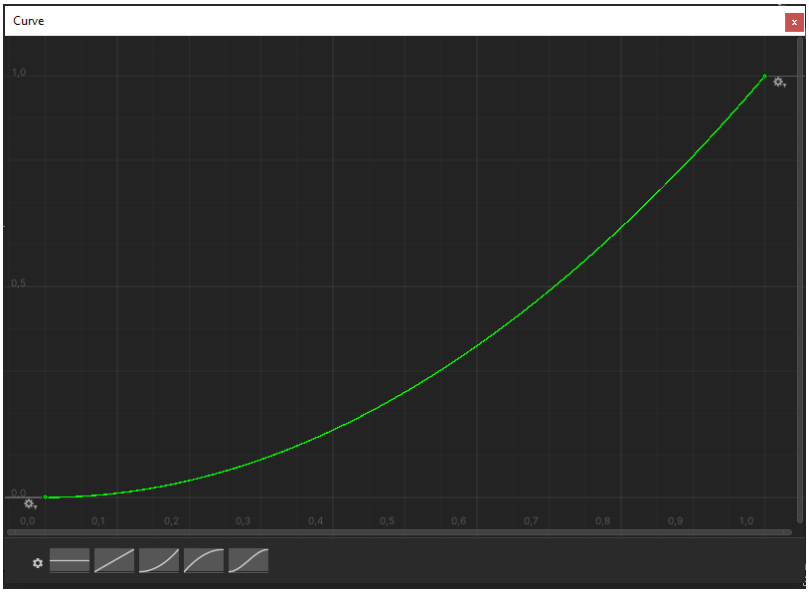


Рисунок 4.5 – Крива анімації здоров’я (рисунок створений самостійно)

У класі EnemySpawner реалізовано механізм масштабування супротивників залежно від поточних очок темної енергії. Кожен раз, коли потрібно створити нового супротивника, його характеристики та деякі характеристики спавну оновлюються відповідно до поточних значень очок темної енергії:

```
private IEnumerator SpawnEnemiesRoutine()
{
    while (true)
    {
        var randomOffset = Random.insideUnitSphere * spawnRadius;
        randomOffset.y = 0f; // Keep the enemies on the same level as
the player
```

```
            var spawnPosition = playerTransform.position + randomOffset;
            if (NavMesh.SamplePosition(spawnPosition, out var hit,
spawnRadius, NavMesh.AllAreas))
            {
                var randomEnemySO = enemies[Random.Range(0,
enemies.Count)];
                var scaledEnemySO =
randomEnemySO.ScaleUpForPoints(scaling, pointsManager.darkEnergyPoints);
                var enemyInstance =
Instantiate(scaledEnemySO.Prefab.gameObject, hit.position,
Quaternion.identity);

                // Setup enemy with ScriptableObject configurations

scaledEnemySO.SetupEnemy(enemyInstance.GetComponent<Enemy>());

                if (enemyInstance.TryGetComponent<EnemyMovement>(out var
enemyMovement))
                {
                    enemyMovement.Player = playerTransform;
                }

                if (enemyInstance.TryGetComponent<NavMeshAgent>(out var
navMeshAgent))
                {
                    navMeshAgent.enabled = true;
                }
            }
            else
            {
                Debug.LogWarning("Unable to find valid NavMesh position for
spawning enemy.");
            }
            yield return new WaitForSeconds(spawnInterval);
        }
```

Ось які характеристики та код, за допомогою якого вони будуть змінюватися:

```
        scaledUpConfiguration.Damage = Mathf.FloorToInt(attack.Damage *
scaling.DamageCurve.Evaluate(darkEnergyPoints));
        scaledUpEnemy.Health = Mathf.FloorToInt(enemy.Health *
scaling.HealthCurve.Evaluate(darkEnergyPoints));
        scaledUpEnemy.Speed = enemy.Speed *
scaling.SpeedCurve.Evaluate(darkEnergyPoints);
        SpawnDelay = InitialSpawnDelay *
Scaling.SpawnRateCurve.Evaluate(darkEnergyPoints + 1);
```

Таким чином, у грі використовується система накопичення очок темної енергії, які безпосередньо впливають на складність супротивників. Це дозволяє динамічно підвищувати рівень виклику для гравця залежно від його прогресу

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування ігрового застосунку

У даному проекті тестування здійснювалось за завчасно підготовленому тест-плану, який був створений за допомогою Mind Map. Тест-план наведено у додатку Г.

Для тестування використовувалися методологія Blackbox, або метод «чорної скриньки», що передбачає тестування гри без знання її внутрішнього коду [13].

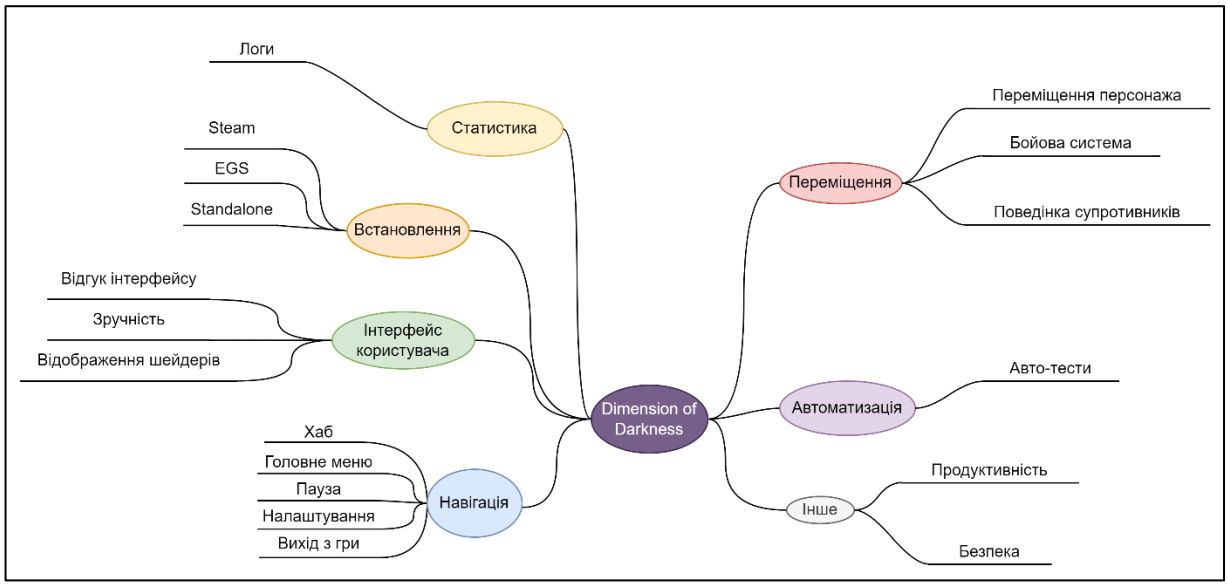


Рисунок 5.1 – Mind Map для тестування гри (рисунок зроблений самостійно)

Ця карта спрощує процес тестування, виділяючи кілька основних компонентів, які повинні працювати разом безперервно, а саме:

а) Автоматизація.

б) Переміщення.

в) Навігація.

г) Встановлення.

д) Статистика.

е) Інтерфейс користувача.

ж) Інше.

Такий підхід тестування дозволив структуровано підійти до цього процесу, що забезпечило зосередження на результатах кінцевого продукту.

5.2 Виявлені помилки

Під час розробки ігрового додатка було зафіксовано 10 помилок, які були включені до списку баг-репортів у вигляді структурованої таблиці. Кожен баг-репорт був розглянутий з точки зору пріоритету від «P1 Високий» до «P3 Низький» та серйозність від «S1 Блокуючий» до «S5 Тривіальний». Крім того, для кожного баг-репорту вказано назва, короткий опис проблеми та компонент програми, який тестується. Для кращого розуміння очікуваного результату та фактичного виходу були додані відповідні стовпці, а також детальний опис кроків, що призвели до помилки. Крім того, до кожного баг-репорту було додано файл, що містить зображення проблеми або спосіб її виправлення. Всі ці матеріали були включені до додатку Д.

Усі виявлені помилки були виправлені під час процесу розробки проекту. Особлива увага приділялася помилкам серйозності S2 і вище, оскільки вони могли вплинути на стабільність роботи ігрового додатка. Кожну помилку ретельно проаналізовано, виправлено та перевірено з метою запобігання їх повторного виникнення у майбутньому.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Соціальне впровадження проекту

Для просування проекту були використані платформи Instagram, YouTube, Telegram, Twitter та Facebook. Під час оцінки результатів стало зрозуміло, що найбільш успішними виявилися Telegram і YouTube, тоді як Instagram, Twitter та Facebook показали менш задовільні результати.

Зокрема, на Telegram та YouTube було отримано найбільш активну аудиторію, яка реагувала на контент та взаємодіяла з ним. Наприклад, відео проекту на YouTube зібрали найбільше переглядів та коментарів, що свідчить про великий інтерес до гри серед глядачів. Такий успіх може бути пояснений тим, що YouTube відомий своєю спрямованістю на відеоконтент і має велику аудиторію, яка активно шукає нові відеоігри.

Щодо Instagram, Twitter та Facebook, вони показали менш ефективні результати. На цих платформах було отримано менше взаємодії з аудиторією, менше лайків, коментарів та переглядів. Це може бути пов’язано з тим, що ці платформи менше підходять для розповсюдження ігрового контенту або вимагають більш специфічного підходу до аудиторії.

Також потрібно розглянути створення сторінок на таких платформах як TikTok. Ця платформа є однією з найбільш популярних соціальних мереж, особливо серед молодої аудиторії. Використання TikTok може допомогти набрати велику аудиторію гравців, які будуть активно просувати відео. Крім того, можна створювати свої тренди, що дуже люблять у цій соціальній мережі.

6.2 Плани щодо публікації гри

Щоб запуск і розповсюдження гри був успішним, потрібно вибрати правильні платформи для публікації. Серед усіх варіантів було обрано розглянути найпопулярніші та найефективніші: Steam та Epic Games Store.

Steam є однією з найпопулярніших ігрових платформ у світі. У ній є широкий спектр інструментів для розробників, що дає змогу опублікувати та просувати гру.

У Steam величезна аудиторія та ком'юніті. Завдяки цьому гра точно не

залишиться непоміченою, а гравці допоможуть поширювати інформацію через фанатські групи та обговорення. Водночас там присутня висока конкуренція через велику кількість ігор на платформі. Крім того, Steam бере комісію в розмірі 30% від продажів, а перевірка гри перед публікацією може зайняти довгий час.

Epic Games Store відносно нова платформа для ігор, яка швидко набирає популярності завдяки агресивній політиці маркетингу та вигідними умовами для розробників.

На цій платформі комісія складає всього 12% від продажів, крім того, Epic Games активно підписує ексклюзивні згоди з розробниками, що може забезпечити додаткову підтримку в просуванні та фінансуванні проекту. Однак ця платформа має меншу аудиторію, як і інструменти, які обмежені порівняно зі Steam.

Для досягнення максимального охоплення аудиторії та збільшення прибутку буде розглянута можливість публікації гри на обох платформах. Це дозволить скористатися перевагами кожної з них.

ВИСНОВКИ

Кінцевим продуктом кваліфікаційної роботи стане створена система штучного інтелекту та ворожих персонажів для ігрового застосунку, розробленого на базі Unity, а також демонстраційна версія цього додатку у жанрі shoot’em up у науково-фантастичній естетиці, що призначена для Windows. Гра буде носити назву «Dimension of Darkness».

Протягом процесу створення буде введено систему абстракцій, яка втілює в собі ворожих персонажів та їх здібності. З’являться вороги різних типів, кожен з яких матиме свої унікальні характеристики та поведінкові стратегії. Система буде використовувати інтерфейси для керування ворогами, що спростить додавання нових функцій.

У процесі роботи було здійснено глибокий аналіз спеціалізованої області та наявних програмних продуктів, що допомогло виявити сильні та слабкі сторони цих систем. На підставі аналізу були окреслені шляхи поліпшення існуючих продуктів і визначені ключові вимоги та завдання для розробки нового ігрового додатку.

Для створення продукту були використані ігровий рушій Unity та мова програмування C#. Як середовища розробки обрано Unity Editor та JetBrains Rider.

Інтеграція розробленої системи штучного інтелекту в ігровий застосунок для Windows буде здійснена успішно. Система буде створена з використанням інструментів Unity, що забезпечить ефективне використання можливостей двигуна для розробки складних ігрових систем. Створена система штучного інтелекту сприятиме покращенню ігрового процесу, надаючи гравцям динамічних та адаптивних ворогів.

Розроблений ігровий додаток може бути вдосконалений шляхом додавання локалізації, системи збереження, нових ворогів та їхньої поведінки, а також арсеналу зброї.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wikipedia [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Shoot_%27em_up (дата звернення: 21.04.2024)

2. Postmotreli.su [Електронний ресурс] – URL: https://posmotreli.su/index.php/Shoot_%27em_up (дата звернення: 21.04.2024).

3. Mezha [Електронний ресурс] – URL: https://mezha.media/articles/yak-shi-vykorystovuiut-u-rozrobtsi-videoihor/ (дата звернення: 21.04.2024).

4. Assault Android Cactus+ Steam [Електронний ресурс] – URL: https://store.steampowered.com/app/250110/Assault_Android_Cactus/ (дата звернення: 21.04.2024).

5. Sky Force Reloaded Steam [Електронний ресурс] – URL: https://store.steampowered.com/app/667600/Sky_Force_Reloaded/ (дата звернення: 21.04.2024).

6. Fandom [Електронний ресурс] – URL: https://contra.fandom.com/wiki/Contra_Wiki (дата звернення: 21.04.2024).

7. Wikipedia [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Діаграма_прецедентів (дата звернення: 21.04.2024).

8. Evergreen [Електронний ресурс] – URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 21.04.2024).

9. Flexberry [Електронний ресурс] – URL: https://flexberry.github.io/en/gpg_statechart-diagram.html (дата звернення: 21.04.2024).

10. Unity Documentation [Електронний ресурс] – URL: https://docs.unity3d.com/Manual/com.unity.ai.navigation.html (дата звернення: 21.04.2024).

11. Unity Documentation [Електронний ресурс] – URL: https://docs.unity3d.com/2020.2/Documentation/Manual/AnimationLayers.html (дата звернення: 21.04.2024).

12. Unity Documentation [Електронний ресурс] – URL: https://docs.unity3d.com/Manual/class-ScriptableObject.html (дата звернення:

21.04.2024).

13. Wikipedia [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Тестування_методом_чорної_скриньки (дата звернення: 21.04.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

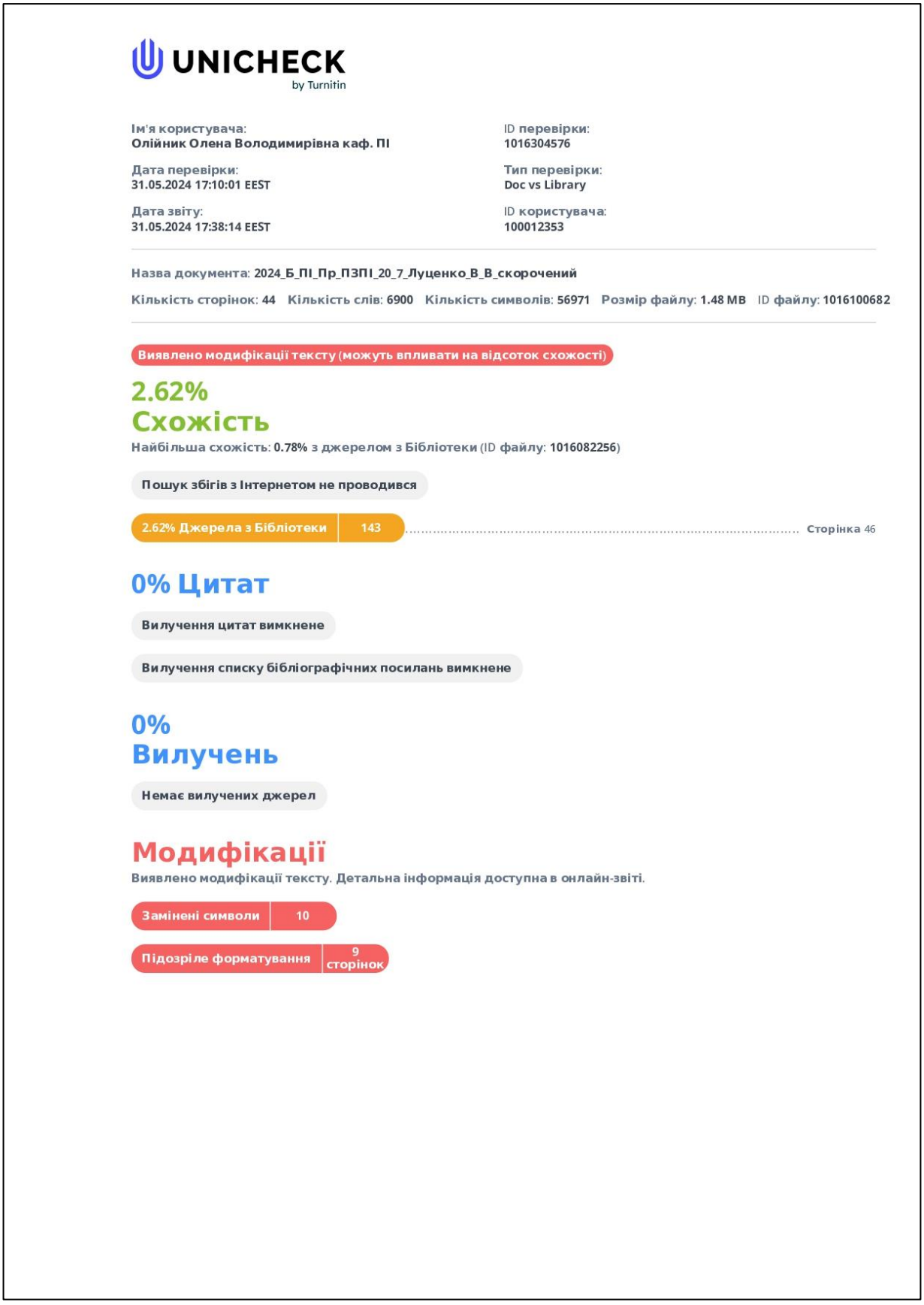


Рисунок А.1 – Звіт Unicheck

ДОДАТОК Б

Слайди презентації

Рисунок Б.1 – Слайд 1

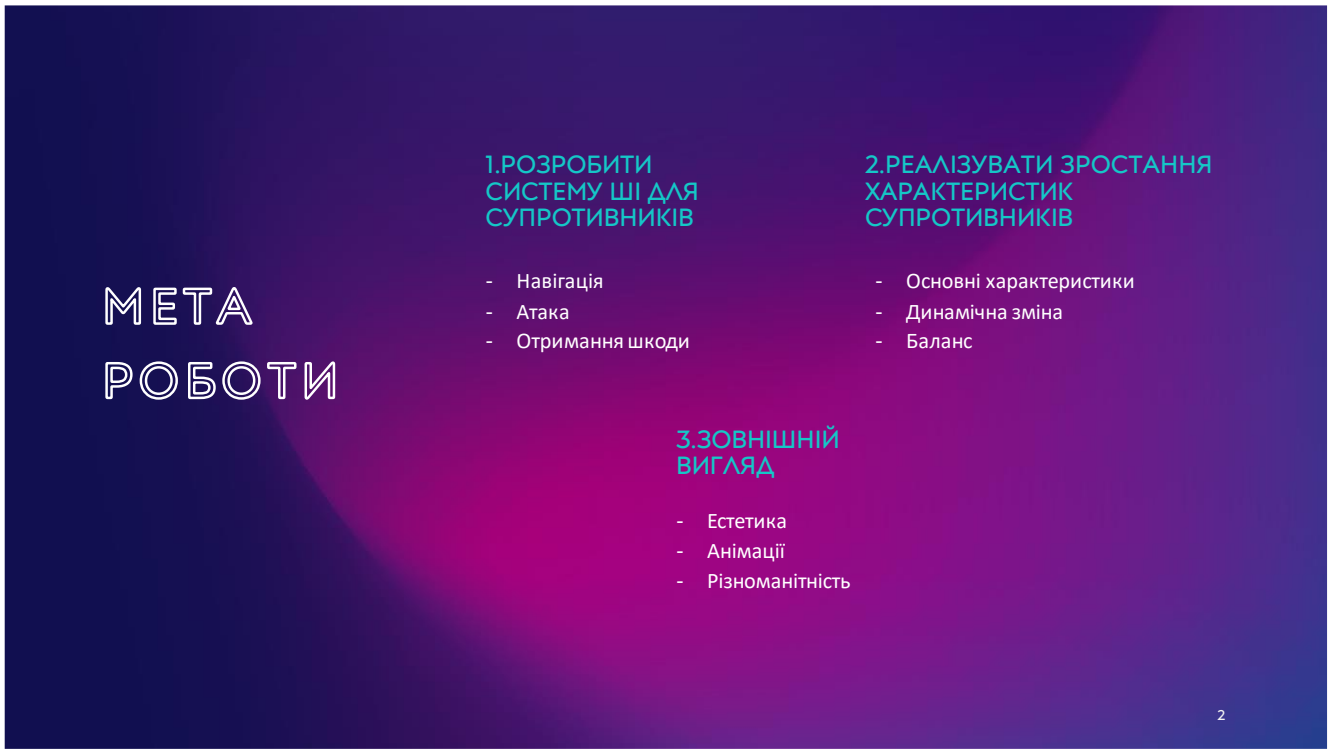


Рисунок Б.2 – Слайд 2

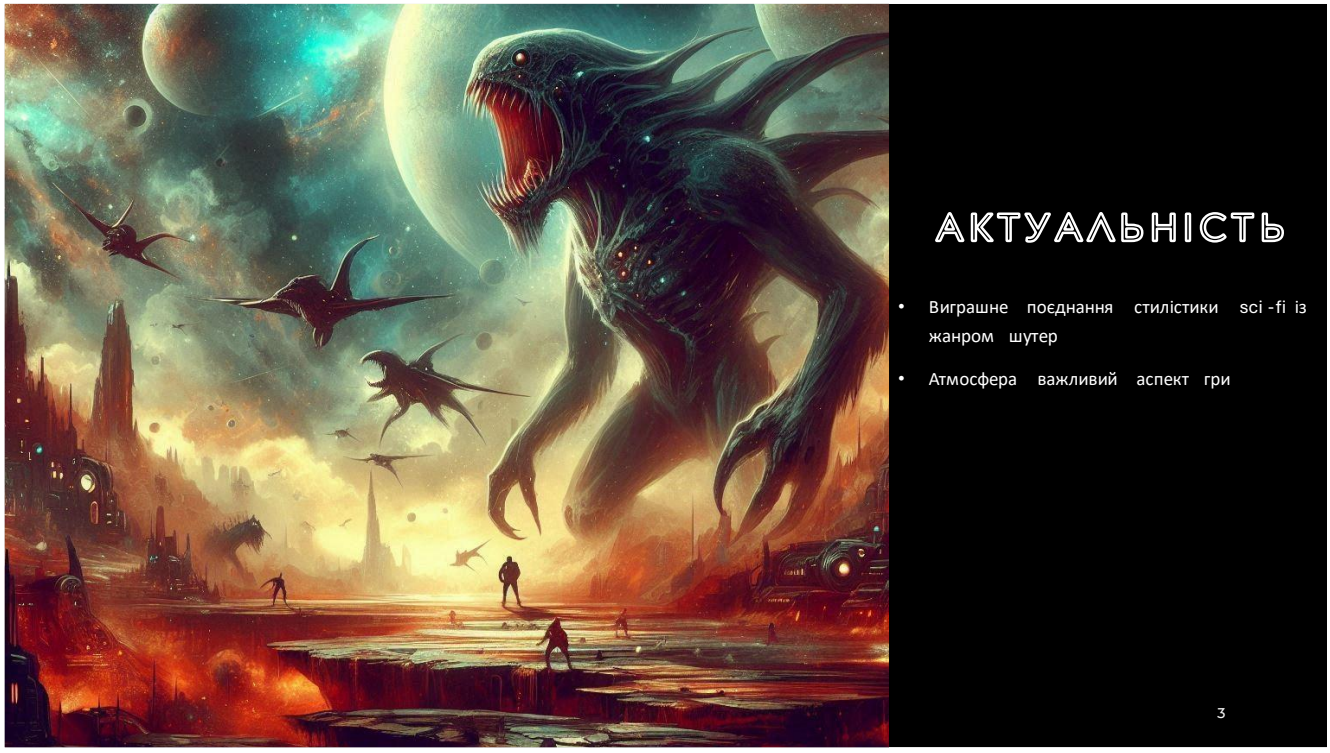


Рисунок Б.3 – Слайд 3

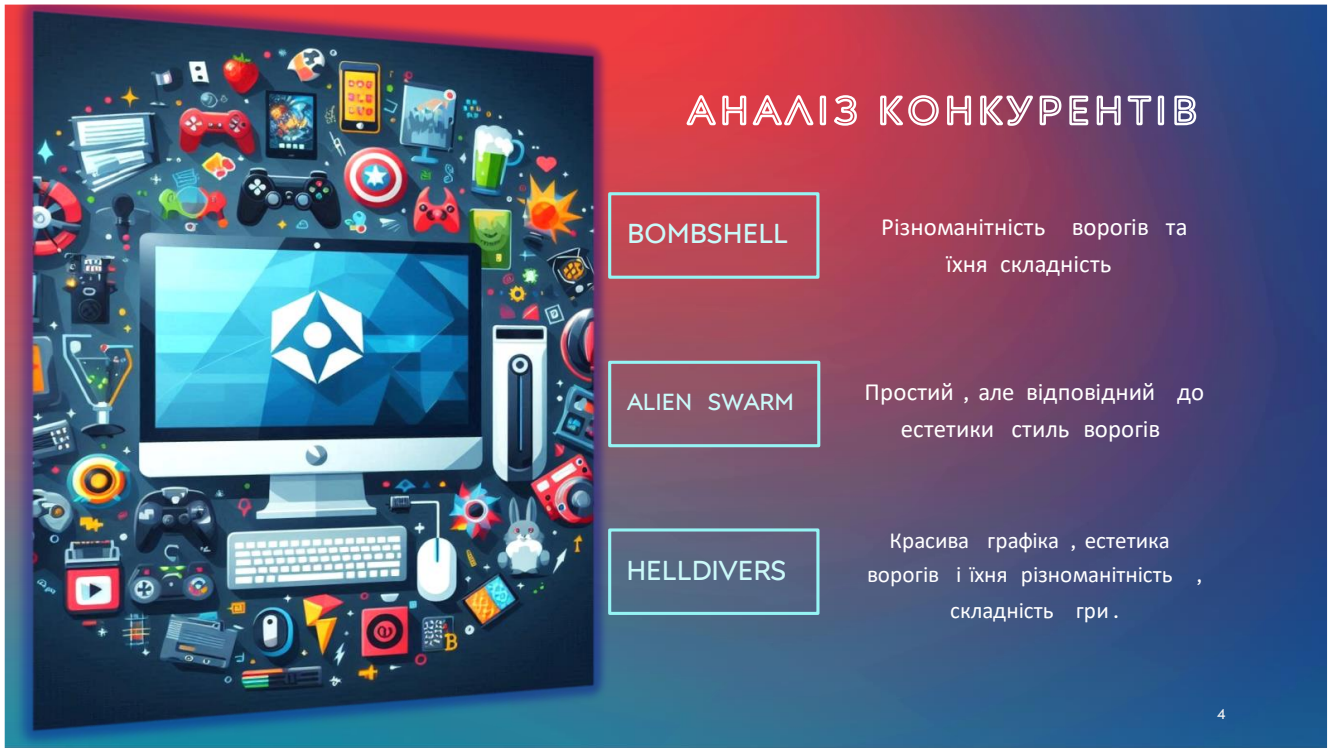


Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

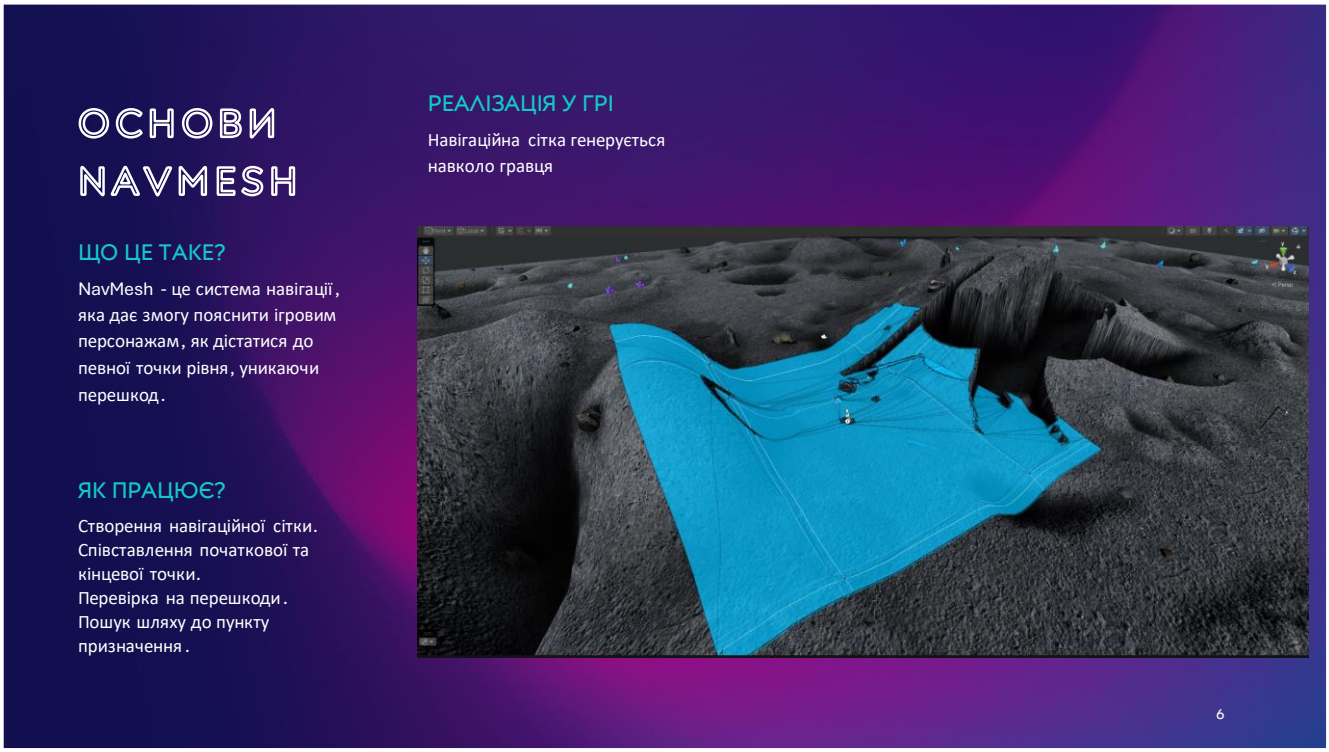


Рисунок Б.6 – Слайд 6

КОД, ЩО ВІДПОВІДАЄ ЗА СТВОРЕННЯ НАВІГАЦІЙНОЇ СІТКИ

```
private void BuildNavMeshes(bool Async)
{
    for (var i = 0; i < Surfaces.Count; i++)
    {
        var surface = Surfaces[i];
        var navMeshData = NavMeshDataList[i];

        var navMeshBounds = new Bounds(Player.transform.position,
NavMeshSize);

        IEnumerable<NavMeshModifier> modifiers =
surface.collectObjects == CollectObjects.Children
            ? GetComponentsInChildren<NavMeshModifier>()
            : NavMeshModifier.activeModifiers;

        var markups = modifiers
            .Where(t => (surface.layerMask & (1 << t.gameObject.layer))
!= 0 &&
                        t.AffectsAgentType(surface.agentTypeID))
            .Select(t => new NavMeshBuildMarkup
            {
                root = t.transform,
                overrideArea = t.overrideArea,
                area = t.area,
                ignoreFromBuild = t.ignoreFromBuild
            }).ToList();
```

```
        if (surface.collectObjects == CollectObjects.Children)
            NavMeshBuilder.CollectSources(transform,
surface.layerMask, surface.useGeometry, surface.defaultArea,
                markups, Sources);
        else
            NavMeshBuilder.CollectSources(navMeshBounds,
surface.layerMask, surface.useGeometry,
                surface.defaultArea, markups, Sources);

        Sources.RemoveAll(source =>
            source.component != null &&
source.component.gameObject.GetComponent<NavMeshAgent>()
!= null);

        if (Async)
            NavMeshBuilder.UpdateNavMeshDataAsync(navMeshData,
surface.GetBuildSettings(), Sources,
                new Bounds(Player.transform.position, NavMeshSize));
        else
            NavMeshBuilder.UpdateNavMeshData(navMeshData,
surface.GetBuildSettings(), Sources,
                new Bounds(Player.transform.position, NavMeshSize));
    }
}
```

```
private IEnumerator CheckPlayerMovement()
{
    var Wait = new WaitForSeconds(UpdateRate);

    while (true)
    {
        if (Vector3.Distance(WorldAnchor, Player.transform.position) >
MovementThreshold)
        {
            BuildNavMeshes(true);
            WorldAnchor = Player.transform.position;
        }

        yield return Wait;
    }
}
```

```
private void Start()
{
    NavMeshDataList = new List<NavMeshData>();

    foreach (var surface in Surfaces)
    {
        var navMeshData = new NavMeshData();
        NavMesh.AddNavMeshData(navMeshData);
        NavMeshDataList.Add(navMeshData);
    }

    BuildNavMeshes(false);
    StartCoroutine(CheckPlayerMovement());
}
```

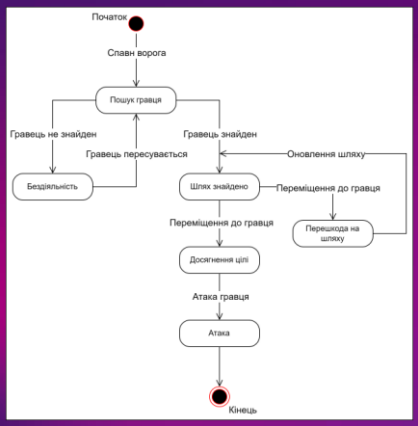
7

Рисунок Б.7 – Слайд 7

ПЕРЕСУВАННЯ ТА АТАКА СУПРОТИВНИКІВ

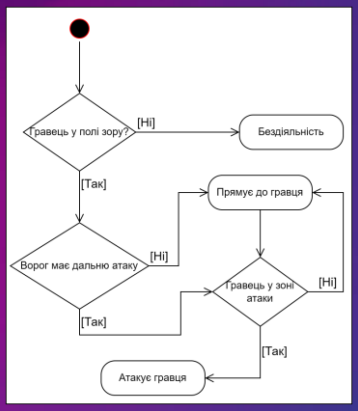
ПЕРЕСУВАННЯ

Кожен ворог отримує ціль (гравця), після чого йде пошук шляху за допомогою NavMesh.



АТАКА

У кожного ворога є радіус атаки. Коли гравець потрапляє в цей радіус, ворог отримує команду атакувати.



8

Рисунок Б.8 – Слайд 8

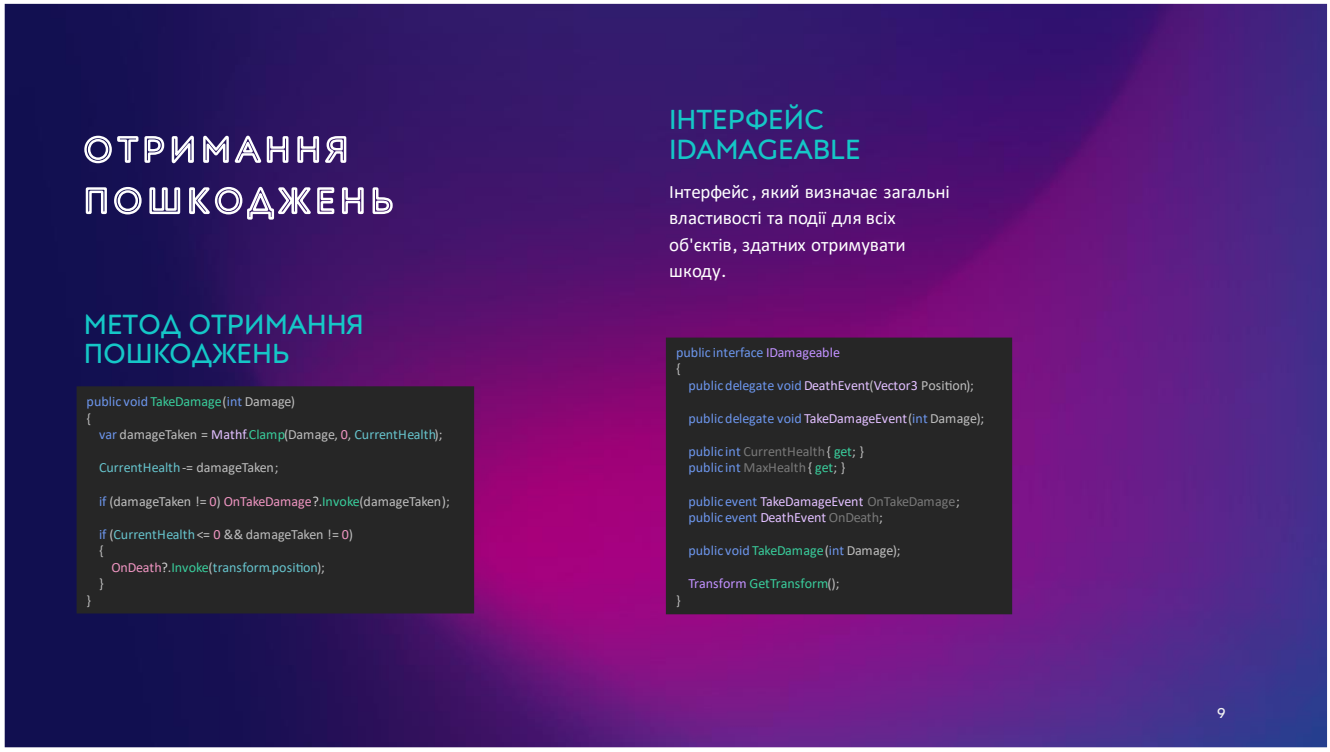


Рисунок Б.9 – Слайд 9

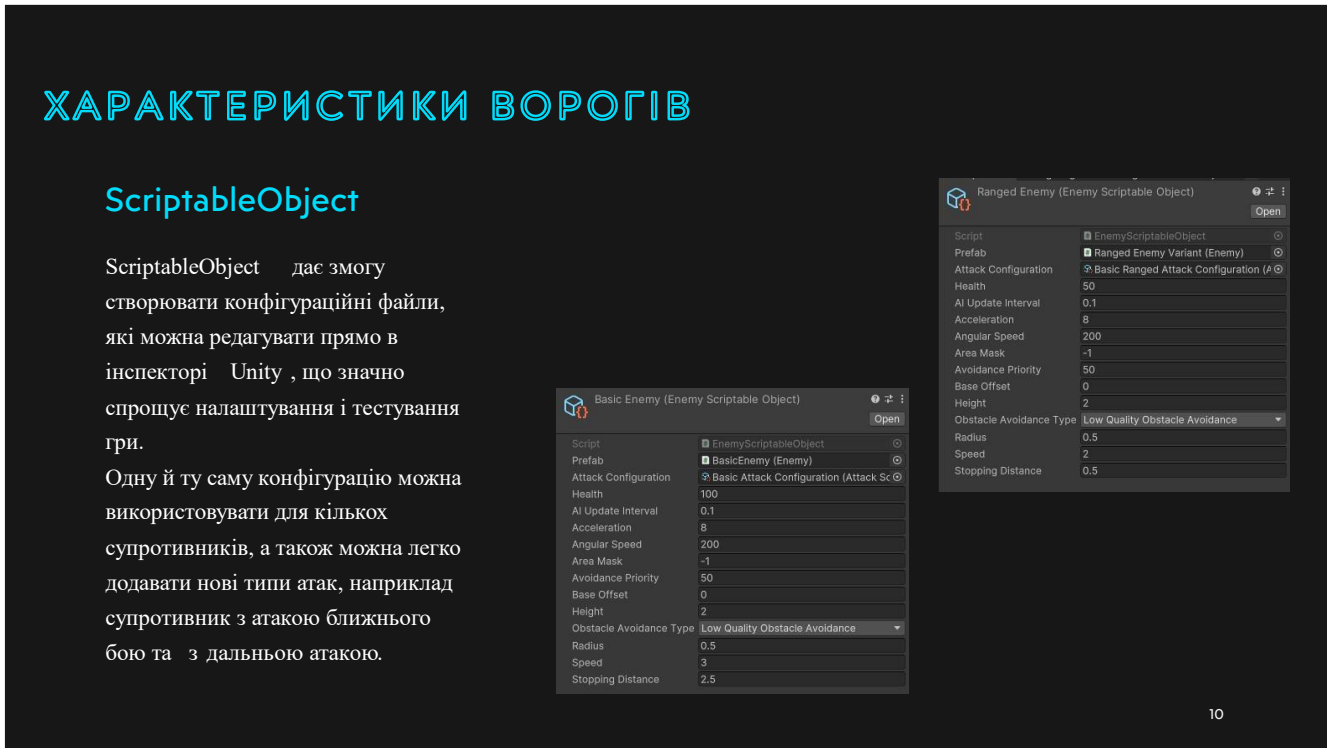


Рисунок Б.10 – Слайд 10

Рисунок Б.11 – Слайд 11

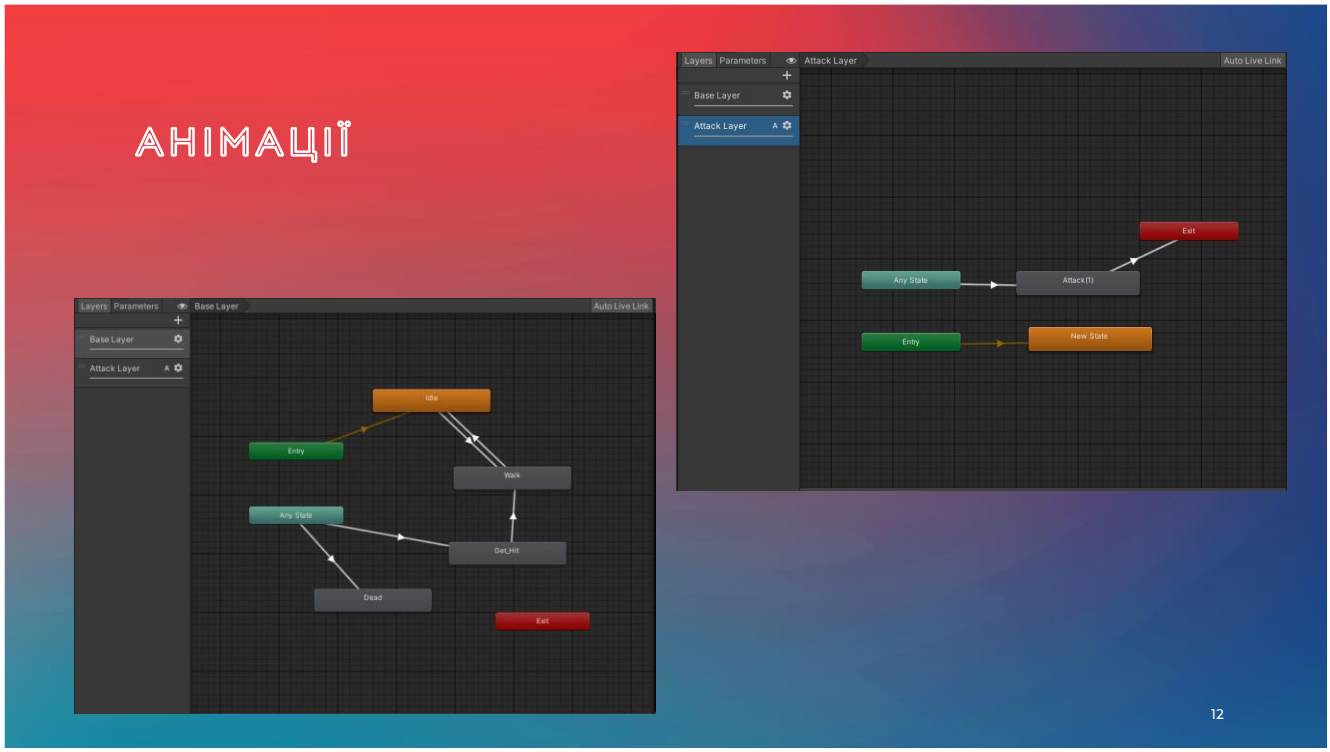


Рисунок Б.12 – Слайд 12

Рисунок Б.13 – Слайд 13

ДОДАТОК В

Концепт-документ до гри

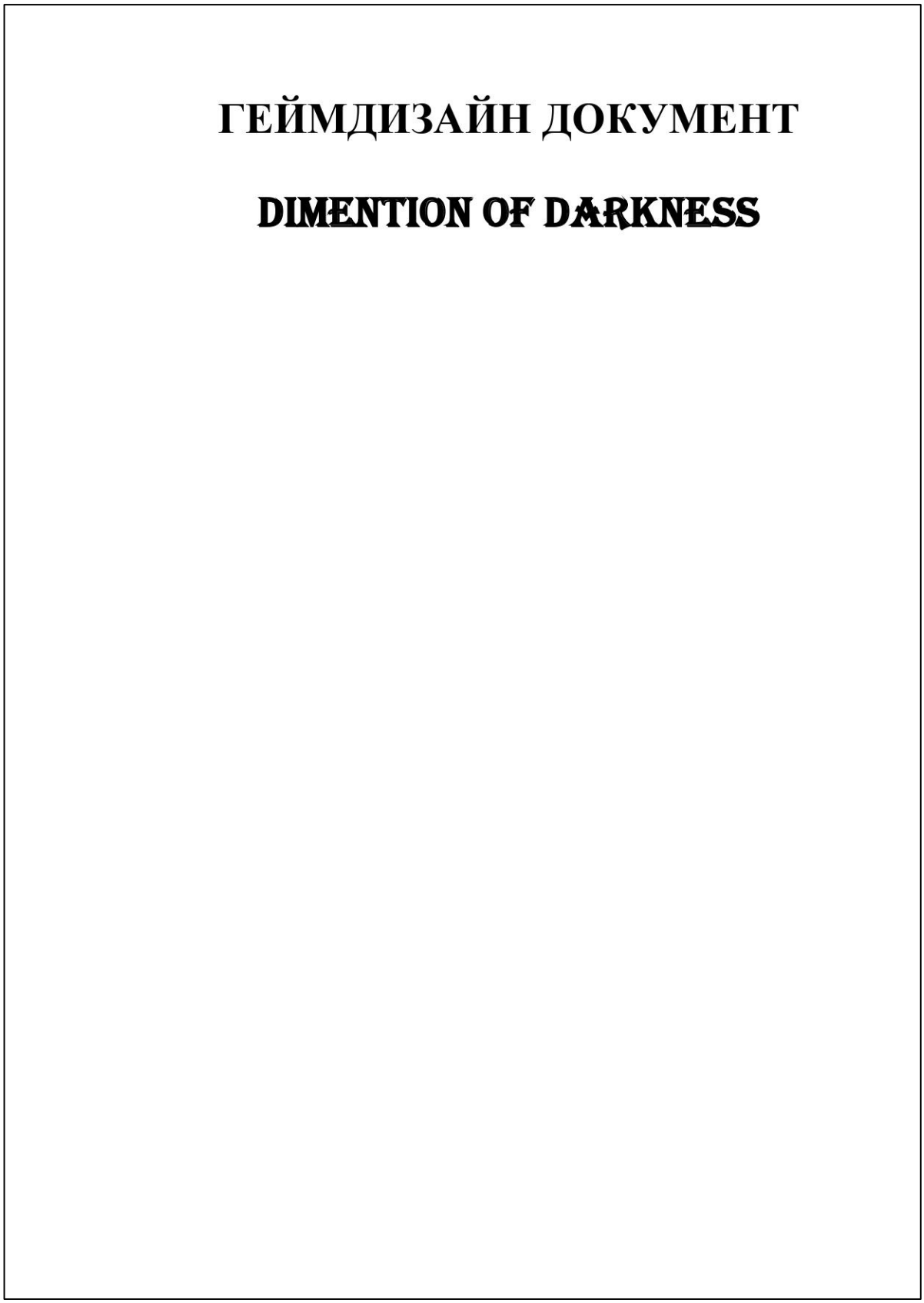


Рисунок В.1 – Сторінка 1 концепт-документу до гри

1 СПЕЦИФІКАЦІЯ

1.1 Тетра

Механіка: Shoot`em-up

Технологія: PC

Історія: ГГ-науковець

Естетика: Fantastic + Darkness

1.2 Цільова аудіторія

Головною аудиторією нашої будуть чоловіки 16-40 років. Цю аудиторію мають привабити: динамічна система бою, графічні ефекти зброї, кооператив, можливість безмежно розвивати персонажа.

1.3 USP

- Спробуйте себе у ролі відчайдушного героя у невідомому темному вимірюванні, який намагається повернутись до дому;
- отримайте задоволення від динамічної бойової системи, знищуючи ворогів;
- не майте обмежень по рівню розвинення персонажа;
- безмежно знищуйте ворогів їдучих постійним натиском на вас.

1.4 Час ігрової сесії

Гра не має рівнів або місій тощо. Гра складається із забігів. Кожен забіг, на початковій стадії гри, може продовжуватись від 5 до 30 хвилин. З подальшим прогресом гравця середній час забігу буде збільшуватись.

Рисунок В.2 – Сторінка 2 концепт-документу до гри

2 ГЕЙМПЛЕЙ

Головною метою гравця за гру буде назбирати достатню кількість енергії, за один забіг, щоб відремонтувати пристрій який допоможе повернутись нашому герою додому. Гравцю треба назбирати необхідну кількість енергії саме за один забіг, бо поміж забігами енергія накопичуватись не буде та при старті нового забігу стара енергія знищується. Це і є основне випробування для гравця. Також йому буде заважати збирати енергію постійний натиск потвор. Щоб мати змогу назбирати достатньо енергії за один раз, гравцю треба зробити багато забігів щоб вдосконалити зброю костюм тощо.

Гра ділиться на два етапи: забіг та відпочинок на базі. Забіг є основним етапом під час якого гравець повинен отримати основне задоволення від гри. Під час відпочинку на базі гравець має змогу: витратити зібрані ресурси на вдосконалення спорядження, відпочити та зберегти гру.

Рисунок В.3 – Сторінка 3 концепт-документу до гри

3 ІСТОРІЯ ТА СЮЖЕТ

3.1 Вступ

Гра не є сюжетно орієнтованою, але гра має історію та мінімальний сюжет щоб гравець розумів логіку та причини того що відбувається під час гри.

3.2 Історія

2087 рік, вчені у науково-дослідницькому центрі досліджують паралельні вимірювання та можливості із ними взаємодіяти. Одного разу 32-річному вченому вдається винайти пристрій який може розривати простір та створювати стабільний портал із іншим вимірюванням, але він навіть не має здогадів що можна очікувати від того вимірювання. Цікавість бере верх над групою вчених та вони віршують розпочати експеримент по запуску зв'язуючого пристрою, не отримавши на це дозвіл від керівництва та не попередивши інших вчених.

Десяте листопада 2087 року, зв'язуючий пристрій під назвою PSDC (Possibly a Stable Dimensional Connector) не справляється з енергетичним навантаженням та бокові опори які тримали портал ламаються, портал одразу збільшується у розмірі та доходячи до розмірів усієї лабораторії, PSDC вимикається через перенавантаження, в наслідок чого лабораторія опиняється у іншому вимірювання. Дослідники мають деяке обладнання та генератор у лабораторії що дозволить їм прожити деякий час. Дослідникам треба відремонтувати PSDC тому вони вирішують відправитись за межі лабораторії для досліджень та вирішення проблеми.

Під час першого виходу з лабораторії, Уолтер, перший дослідник доброволець, бачить перед собою всюди тьму та густий також темний туман. Трохи озирнувшись Уолтер побачить дивне створіння схоже чи на павука чи на людину та воно вочевидь було налаштоване вороже. Зреагувавши Уолтер вбиває це створіння із пістолета, який йому видали. Трохи дослідивши довкілля, він розуміє, завдяки його спорядженню він

Рисунок В.4 – Сторінка 4 концепт-документу до гри

може збирати з довкілля та створінь що тут живуть енергію, це допоможе вченим відремонтувати PSDC та повернутись до дому. Також потвори що тут живуть, як помітив Уолтер, створені із різних матеріалів що дозволяє використовувати їх залишки як матеріал для АНТ (atomic handheld transducer), це пристрій який є майже у кожного у 2087 році, він може розбирати речі на окремі атоми та створювати з них те що хоче власник пристрою.

Повернувшись до лабораторії Уолтер помічає що ємності його рюкзаку для збору енергії недостатньо щоб назбирати енергії за раз. Але що набагато гірше Уолтер розуміє темна енергія яку він назбирав дуже швидко анігілює з тою енергією що телепортувалась в це вимірювання. Через це вчені не мають змоги просто накопичувати енергію, значить їм потрібно вдосконалити рюкзак Уолтера щоб він міг принести багато енергії за раз.

Подальшою метою Уолтера є зібрати достатню кількість ресурсів щоб вдосконалити своє обладнання та відремонтувати PSDC щоб повернутись додому, але створіння що живуть у цьому вимірюванні не дадуть йому це так просто зробити.

Рисунок В.5 – Сторінка 5 концепт-документу до гри

4 МЕХАНІКИ

4.1 Механіки гравця

Поява (у забігу)

Для того, щоб почати забіг, гравцеві (усім гравцям) необхідно зайти в телепорт. Після того почнеться сам забіг.

Поява (на базі)

Після того, як гравець у забігу загине, він опиниться на базі. Кількість твердотільного матеріалу скидається до 0.

Ривок та стрибок

Ривок дозволяє гравцеві практично моментально переміститися на певну дистанцію у напрямку руху гравця. Гравець може зробити ривок, натиснувши на Space. Ривок витрачатиме певну кількість витривалості.

Отримання твердотільного матеріалу та темної енергії з убитих супротивників

Після смерті противника, на рахунок гравця переходитиме певна кількість темної енергії, а також із противника може випасти якась кількість твердотільного матеріалу, з якого гравець зможе створити нові предмети.

Рисунок В.6 – Сторінка 6 концепт-документу до гри

4.2 Механіки супротивника

Спавн

Вороги з'являтимуться навколо гравця, поза його полем зору. Частота появи противників залежить від часового інтервалу, а також від просування гравця по карті. Також будуть додаватися/мінятись типи противників, це залежатиме від просування гравця по карті та кількості накопичених одиниць темної енергії під час забігу.

Атака

Вороги зможуть використовувати різні види атак, такі як удари у ближньому бою, стрілянина зі зброї, кидання снарядів, магічні атаки тощо. Атаки деяких супротивників матимуть певний патерн. Залежно від типу ворога, буде змінюватися швидкість/збиток атаки. Також існуватимуть особливі атаки, наприклад атаки по області, уповільнення, зменшення огляду тощо.

Смерть

Смерть ворога супроводжуватиметься анімацією та/або візуальними ефектами. Після смерті противника на їхньому місці з'являтиметься ресурс, який гравець може підібрати та використати надалі. Також смерть противників може супроводжуватися певними особливими ефектами смерті, наприклад вибухом.

Рисунок В.7 – Сторінка 7 концепт-документу до гри

4.3 Механіки світла та темряви

Механіка освітлення

Направляючи світло на туман, гравець буде розсіювати його, поки не перенаправить світло в іншу область. У освітленій області не можуть сповнитися супротивники. Для того, щоб висвітлити якусь область, існують певні предмети, що витрачаються, або поліпшення.

Механіка туману

У всіх місцях карти, які гравець ніяк не висвітлює, він бачитиме чорний туман. Навіть якщо гравець одного разу вже висвітлив область, то, переставши висвітлювати область, до неї повернеться туман.

4.4 Механіки зброї

Перезаряджання

Практично у всієї зброї в грі є свій час на перезарядку після пострілу, при цьому найчастіше зброя уповільнюватиме гравця під час перезарядки, а такі рухи як біг або стрибки через перешкоди збиватимуть таймер перезарядки, повертаючи його до початку відліку.

Заряд пострілу

Деяка зброя у грі завдає постріли після процесу заряду певної тривалості. Заряд здійснюється затисканням ЛКМ, після того, як гравець відпустить ЛКМ зброю зробить удар/постріл. Більшість зброї у грі з подібною механікою матиме максимальну межу заряду для удару/пострілу. Чим більшої потужності заряд, тим більшої сили буде постріл.

Рисунок В.8 – Сторінка 8 концепт-документу до гри

Простріл противників

Деяка зброя у грі прострілюватиме супротивників наскрізь, наприклад, така зброя, як снайперська гвинтівка. При пострілі, куля з такої зброї пробиватиме всіх супротивників на своєму шляху, зупиняючись (зникаючи) на певній відстані.

4.5 Механіки костюму персонажа

Загальні відомості

Енергетичний щит

Костюм має вбудований енергетичний щит, він має певну кількість очок, які будуть забиратися при отриманні збитків від супротивників. Після того, як щит буде розряджений, гравець почне втрачати очки здоров'я від отриманої шкоди. Щит починає автоматично поступово заряджатися до максимального заряду поки гравець не отримує шкоди від супротивників певний час. Отримання втрат під час заряду щита зіб'є процес зарядки, кількість очок зберегтися на тому рівні, де зупинилася зарядка.

Рисунок В.9 – Сторінка 9 концепт-документу до гри

5 ІНТЕРФЕС КОРИСТУВАЧА

5.1 HUD

На екрані гравець бачить:

- рівень свого здоров'я, витривалості та енергощита костюма;
- слоти зі зброєю та обрану зброю, а також максимальну та фактичну кількість патронів у магазині.;
- фактичну та необхідну для підвищення рівня гри кількість темної енергії.

5.2 Система контролю

Гравець має доступ до наступного контролю:

- Переміщення – [W][A][S][D];
- оглянутись навколо – миша;
- ривок/перестрибнути перешкоду – [Space];
- біг – [Shift];
- постріл/прицілювання – [LMB]/[RMB];
- особлива атака зброї – [Q];
- взаємодія із інтерактивними предметами – [E];
- меню паузи – [Esc].

5.3 Аудіо ефекти

Гра буде музичний супровід. Звукові ефекти від пострілів та заряду зброї, противники та персонаж також будуть мати свої звукові ефекти. А також деякі предмети навколо будуть створювати звукові ефекти.

Рисунок В.10 – Сторінка 10 концепт-документу до гри

6 РЕФЕРЕНСИ

6.1 Головні герої

Усі виглядають приблизно однаково. Білий костюм-екзоскелет, зріст десь 1.8м

Рисунок В.11 – Сторінка 11 концепт-документу до гри

Рисунок В.12 – Сторінка 12 концепт-документу до гри

Рисунок В.13 – Сторінка 13 концепт-документу до гри

6.2 Противники

Повністю чорні створіння, місяцями схожі на людей, місцями на тварин. Замість очей та роту чорні впадини.

Рисунок В.14 – Сторінка 14 концепт-документу до гри

Рисунок В.15 – Сторінка 15 концепт-документу до гри

Рисунок В.16 – Сторінка 16 концепт-документу до гри

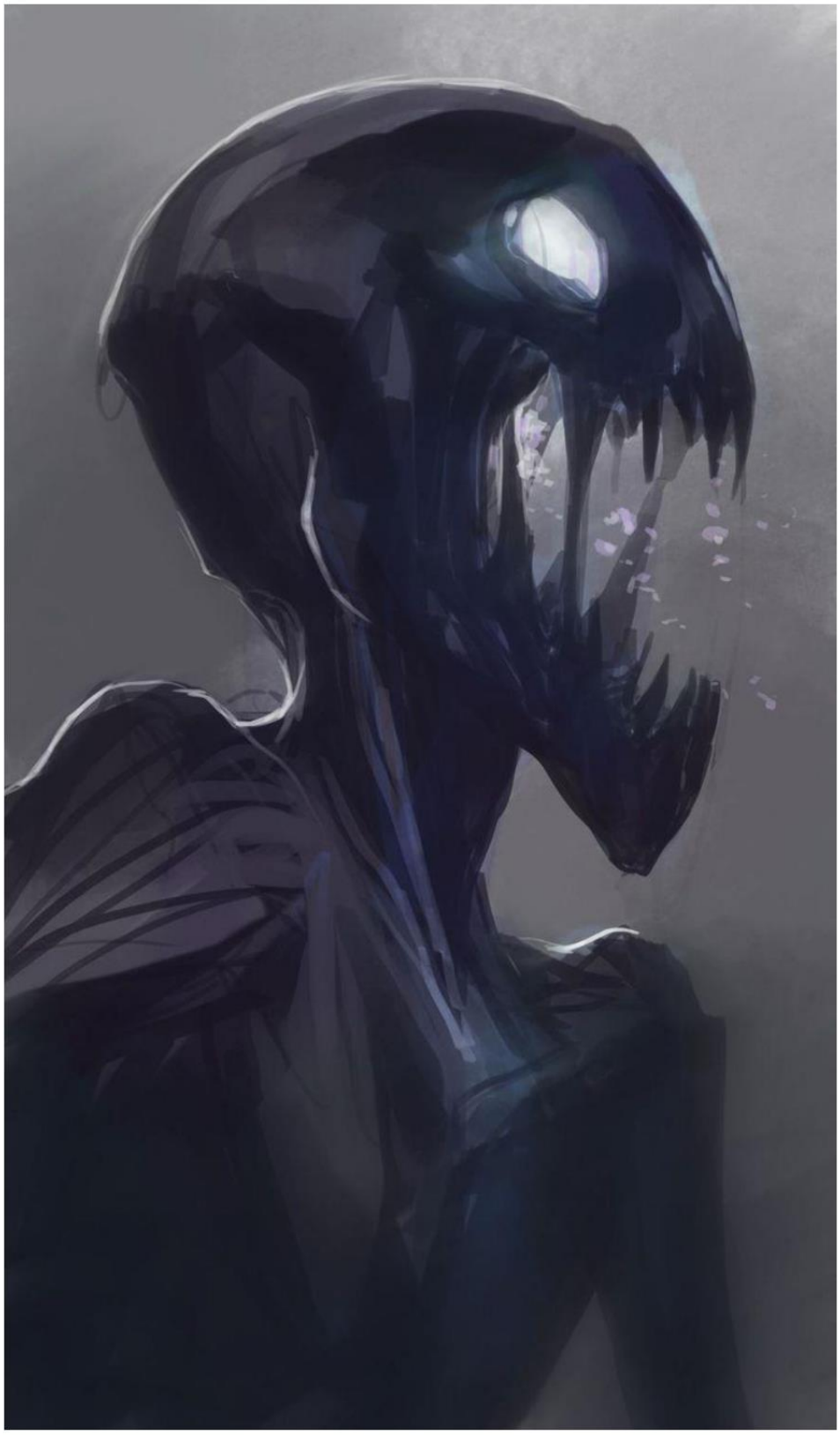


Рисунок В.17 – Сторінка 17 концепт-документу до гри

6.3 Біом

Перший біом це звичайна земля рожево-коричневого кольору. По краям біомів будуть розломи, поламані мости, автомобілі та інше. Вид згори.

Рисунок В.18 – Сторінка 18 концепт-документу до гри

Рисунок В.19 – Сторінка 19 концепт-документу до гри

6.4 Лабораторія

Приміщення з великою кількістю поламаної техніки, такої як комп’ютери, прінтери, якісь колби і так далі. В лабораторії будуть місцями горстки землі, із полу стирчатимуть сталагміти. Задня частина лабораторії буде завалена та зливатиметься із землею. Тільки вид згори.

Рисунок В.20 – Сторінка 20 концепт-документу до гри

Рисунок В.21 – Сторінка 21 концепт-документу до гри

Рисунок В.22 – Сторінка 22 концепт-документу до гри

6.5 Туман

Туман буде всюди, окрім шляху де світитиме ліхтар, або освітлювальні предмети.

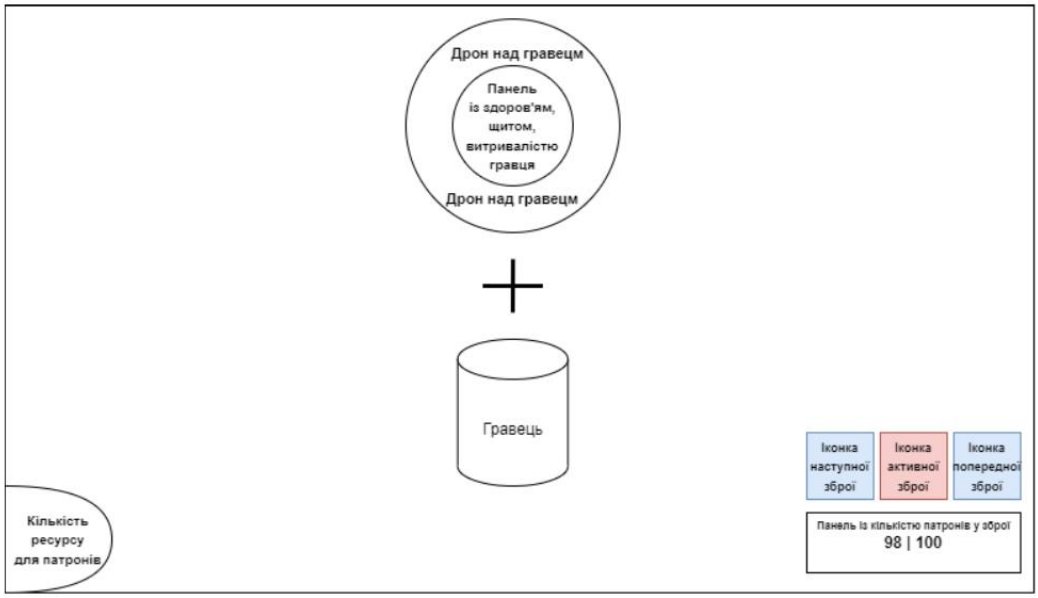
6.6 Головне меню

Головне меню буде складатися із фону, та кнопок «Одиночна гра», «Мультиплеєр», «Налаштування» та «Вихід».

Рисунок В.23 – Сторінка 23 концепт-документу до гри

6.7 Меню налаштувань

6.8 Ігровий інтерфейс



7 РОЗГОРТАННЯ ТА РОЗРОБКА

7.1 Порядок встановлення

Рисунок В.24 – Сторінка 24 концепт-документу до гри

Гру можливо буде інсталювати через інсталятор або через Steam/EGS

7.2 Системні вимоги

Операційна система: Windows 8 або кращє.

Процесор: Intel 2.77GHz Quad-core.

ОЗП: 8 Gb.

Відеокарта: NVIDIA GTX 550 Ti.

DirectX: 10.

Місце на диску: до 1Gb.

8 РОЗРОБКА

Гра буде розроблена на рушії Unity.

Репозиторій: https://github.com/Dayman267/ZombieSurviveSmash1.

Рисунок В.25 – Сторінка 25 концепт-документу до гри

ДОДАТОК Г

План тестування

Назва проекту

Dimension of Darkness

План випробувань і тестування

Test Plan

Версія 1.0

Version 1.0

Рисунок Г.1 – Сторінка 1 плану тестування

Історія змін (Revision History)

Дата	Версія	Детальний опис	ПІБ Автора
18.03.2024	1.0	Створення тест-плану	Попов Д.М.
			Хом’яков К.І.
			Луценко В.В.

Рисунок Г.2 – Сторінка 2 плану тестування

1. ВСТУП

1.1 Мета

Метою складання даного тест плану є опис процесу тестування гри «Dimension of Darkness». Мета документу - координація роботи процесу розробки у сфері контролю якості продукту. Документ призначений групі тестування для ознайомлення з характером майбутніх робіт, аналізу і розбиття на підзадачі. Документ дозволяє отримати уявлення про заходи з тестування проекту

1.2 Довідкова інформація

Уолтер - головний герой, який опинився у паралельному вимірі після невдалого експерименту. У процесі дослідження цього виміру він виявляє темну енергію, яка може допомогти йому повернутися додому. Упродовж гри гравець досліджує нове оточення, бореться з небезпечними істотами та збирає ресурси для повернення додому, але він має подолати безліч перешкод, щоб досягти своєї мети.

«Dimension of Darkness» – гра у жанрі 2.5D shoot’em up з елементами Roguelike, розроблена для операційної системи Windows. Поки наявна лише англійська локалізація.

1.3 Галузь застосування

Тестування необхідне для досягнення визначених критеріїв якості, що визначені у тестовому плані. Основні мети тестування включають:

- функціональне тестуваня;
- підтвердження якості та надійності програмного забезпечення;
- покращення зручності використання графічного інтерфейсу користувача;
- оптимізацію програмного продукту.

1.4 Визначення проекту

Документ	Створено або доступно	Отримано або перевірено	Автор або ресурс	Примітки
Дизайн-документ гри	Так	Так	Попов Д.М. Хом’яков К.І. Луценко В.В.	

Рисунок Г.3 – Сторінка 3 плану тестування

2. ВИМОГИ ДО ТЕСТУВАННЯ

Функціональне тестування:

- Переміщення персонажа;
- Система освітлення;
- Бойова система;
- Система генерування карти;
- Поведінка супротивників.

Тестування інтерфейсу користувача:

- Перевірка часу відгуку на взаємодії користувача з інтерфейсом;
- Перевірка правильності відображення текстур на персонажах та локаціях.

Тестування встановлення:

- Перевірка наявності гри, що встановлюється.

Рисунок Г.4 – Сторінка 4 плану тестування

3. СТРАТЕГІЯ ТЕСТУВАННЯ

3.1 Типи тестування

3.1.1 Функціональне тестування

Мета випробування	Протестувати механіки гри
Технічний прийом	Протестувати функції: • Переміщення персонажа; • Система освітлення; • Бойова система; • Система генерування карти; • Поведінка супротивників.
Критерії завершення	Усі випробування були проведені

Функціональне тестування в геймдеві розглядає основний функціонал гри, щоб переконатися, що він працює правильно і відповідає вимогам. Це включає перевірку різних аспектів гри, таких як управління персонажем, взаємодія з об'єктами, виконання завдань та багато іншого. Крім того, функціональне тестування включає в себе перевірку колізій.

3.1.2 Тестування інтерфейсу користувача

Мета випробування	Переконатися, що інтерфейс користувача зручний, текстури та анімації правильно відображаються.
Технічний прийом	Перевірити правильність відображення текстур на персонажах, локаціях. Перевірити зручність користувацького інтерфейсу.
Критерії завершення	Усі випробування були проведені.

Тестування інтерфейсу користувача (UI) в геймдеві є важливою частиною процесу розробки, оскільки від правильної роботи інтерфейсу залежить зручність користування грою та користувацький досвід.

Рисунок Г.5 – Сторінка 5 плану тестування

3.1.3 Тестування встановлення гри

Мета випробування	Переконатися, що гра правильно встановлюється
Технічний прийом	Ручна або автоматична установка. Перевірка наявності гри, що встановлюється
Критерії завершення	Усі випробування були проведені

3.2 Інструменти

	Інструмент	Постачальник	Версія
Контроль версій	GitHub	GitHub, Inc.	2024
Управління проєктами	GitHub Projects	GitHub, Inc.	2024
Відслідковування дефектів	Jira	Atlassian	2024

Рисунок Г.6 – Сторінка 6 плану тестування

4. РЕСУРСИ

4.1 Ролі

Людські ресурси		
Працівник	Рекомендований мінімальний обсяг осіб	Конкретні обов'язки або коментарі
Тест-менеджер, Менеджер з тестування	1	Забезпечує управління наглядом. Обов'язки: - технічна підтримка, - придбання відповідних ресурсів, - забезпечення управлінської звітності
Конструктор тестів	1	Визначення, пріоритетів, і реалізація тестів. Обов'язки: - створення плану тестування, - генерація тестових моделей, - оцінка ефективності тестових зусиль
Тестувальник	1	Виконання тестів. Обов'язки: - виконання тестів, - журнал результатів, - відновлення в журналі реєстрації після помилок, - документ зміни.

4.2 Система

Системні ресурси	
Ресурс	Ім'я або тип
ПК для тестування	Персональний ПК
Репозиторій тестування Посилання	https://github.com/Dayman267/DimensionOfDarkness.git

Рисунок Г.7 – Сторінка 7 плану тестування

5. ЕТАПИ ПРОЄКТУ

Таблиця 5.1 – Етапи проєкту

Етап	Обсяг робіт	Дата початку	Дата закінчення
План випробувань	10	10.03.2024	11.03.2024
Тест - дизайн	15	12.03.2024	13.03.2024
Реалізація випробувань	28	14.03.2024	16.03.2024
Виконання тесту	35	18.03.2024	20.03.2024
Оцінка випробувань	7	22.03.2024	23.03.2024

Рисунок Г.8 – Сторінка 8 плану тестування

6. КІНЦЕВИЙ ПРОДУКТ

6.1 Тестова модель

У ході виконання тестування будуть отримані такі елементи: − план тестування – опис цілей та стратегій тестування, методів реалізації процесу тестування; − тест-кейси – документи, що відповідають примірникам тестового сценарію. Повинні містити: унікальний номер, опис, кроки відтворення, пріоритет, важливість тестового сценарію та очікуваний результат; − баг-репорти – звіти про знайдені недоліки в системі із зазначенням рівня серйозності проблеми.

6.2 Звіти з дефектів

Звіти з дефектів будуть створені з використанням текстового процесору MS Word, систем баг-трекінгу Jira, а також систем управління проектом GitHub Project.

Рисунок Г.9 – Сторінка 9 плану тестування

ДОДАТОК Д

Баг-репорти

Таблиця Д.1 – Помилка 1

Назва багу	Спавн ворогів у повітрі
Короткий опис	Вороги спавняться в повітрі й застрягають там
Компонент додатку	EnemySpawner
Важливість	S2 Висока
Пріоритет	P2 Середній
Кроки відтворення	Гравець піднімається на гору Проходить спавн ворога
Фактичний результат	Ворог спавниться у повітрі і остається там «літати»
Очікуваний результат	Ворог має спавнитися на землі
Прикріплений файл	3.1.mp4

Таблиця Д.2 – Помилка 2

Назва багу	Ворог крутиться після атаки
Короткий опис	Якщо ворог б’є і в цей момент швидко повертається, то після цього він крутитиметься
Компонент додатку	Horror_Mutant (Animator Controller)
Важливість	S3 Середня
Пріоритет	P2 Середній
Кроки відтворення	Коли ворог атакує бігати навколо нього
Фактичний результат	Ворог починає крутитися та пересуватися одночасно
Очікуваний результат	Ворог не крутиться, йде прямо за гравцем
Прикріплений файл	3.2.mp4

Таблиця Д.3 – Помилка 3

Назва багу	Часте сканування карти
Короткий опис	Коли генерується нова частина карти, то відтворюється сканування для NavMesh сітки, що створює сильні лаги
Компонент додатку	Tilemap Generator
Важливість	S1 Блокуюча
Пріоритет	P1 Високий
Кроки відтворення	Пересуватися в будь-який бік, поки не створиться нова територія
Фактичний результат	Постійні зависання
Очікуваний результат	Зависань немає
Прикріплений файл	3.3.mp4

Таблиця Д.4 – Помилка 4

Назва багу	Телепортування супротивників
Короткий опис	Противників телепортує, якщо вони заходять за край сітки NavMesh
Компонент додатку	Area Floor Baker
Важливість	S3 Середня
Пріоритет	P3 Низький
Кроки відтворення	Пересуватися в будь-який бік Ворог на краю сітки Ворога "тягне" край сітки
Фактичний результат	Ворога "тягне" край сітки
Очікуваний результат	Ворог остається на своєму місці
Прикріплений файл	3.4.mp4

Таблиця Д.5 – Помилка 5

Назва багу	Застрявання супротивників
Короткий опис	Супротивники не можуть підійматися під кутом вище за 45 градусів
Компонент додатку	NavMeshAgent
Важливість	S2 Висока
Пріоритет	P2 Середній
Кроки відтворення	Завести ворога у яму Вийти з ями, щоб ворог слідкував за гравцем
Фактичний результат	Ворог застряє та не може вибратися
Очікуваний результат	Ворог вилазить як і гравець
Прикріплений файл	3.5.mp4

Таблиця Д.6 – Помилка 6

Назва багу	Підкидання гравця ворогами
Короткий опис	Коли вороги врізаються і б'ють гравця, його підкидає вгору
Компонент додатку	Rigidbody (Player, enemy)
Важливість	S2 Висока
Пріоритет	P2 Середній
Кроки відтворення	Гравець стоїть на місці Ворог починає бити гравця
Фактичний результат	Гравець піднімається в повітря
Очікуваний результат	Гравець залишається на землі
Прикріплений файл	3.6.mp4

Таблиця Д.7 – Помилка 7

Назва багу	Неправильний рух персонажа по осі Y
Короткий опис	Персонаж парить в повітрі
Компонент додатку	CharacterController
Важливість	S2 Висока
Пріоритет	P2 Середній
Кроки відтворення	Гравець підіймається на гору Гравець починає спускатися з гори
Фактичний результат	Персонаж повільно спускається на землю, ширяючи в повітрі
Очікуваний результат	Персонаж спускається з нормальною гравітацією
Прикріплений файл	3.7.mp4

Таблиця Д.8 – Помилка 8

Назва багу	Неприродні тіні від супротивників.
Короткий опис	Тіні дуже різкі та чорні
Компонент додатку	Light
Важливість	S3 Середня
Пріоритет	P3 Низький
Кроки відтворення	Гравець наводить приціл в бік супротивників
Фактичний результат	Тіні дуже різкі та чорні
Очікуваний результат	Більш реалістичні тіні
Прикріплений файл	3.8.mp4

Таблиця Д.9 – Помилка 9

Назва багу	Зміна в навігації супротивників
Короткий опис	Іноді супротивники перестають переслідувати гравця
Компонент додатку	NavMeshAgent
Важливість	S3 Середня
Пріоритет	P2 Середній
Кроки відтворення	Бігати від супротивників
Фактичний результат	Ворог починає йти від гравця
Очікуваний результат	Ворог слідкує за гравцем
Прикріплений файл	3.9.mp4

Таблиця Д.10 – Помилка 10

Назва багу	Різна висота ландшафту
Короткий опис	Ландшафт створен на різній висоті
Компонент додатку	Terrain
Важливість	S3 Середня
Пріоритет	P2 Середній
Кроки відтворення	Підійти на границю між двома ландшафтами
Фактичний результат	Видно щілину між ландшафтами
Очікуваний результат	Гладкий перехід між ландшафтами
Прикріплений файл	3.10.jpg